SPARSITY EXPLOITATION IN THE EXTENDED NAPHTALI-SANDHOLM METHOD FOR SOLUTION OF INTERLINKED MULTISTAGED SEPARATORS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

MANOJ KUMAR JAIN

to the

DEPARTMENT OF CHEMICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JULY, 1987

CHE-1987-M-JAI-SPA

Thesis
660.2842
J199s

CERTIFICATE

It is certified that the present work entitled, "SPARSITY EXPLOITATION IN THE EXTENDED NAPHTALI-SANDHOLM METHOD FOR SOLUTION OF INTERLINKED MULTISTAGED SEPARATORS" has been carried out by Mr. Manoj Kumar Jain under my supervision and that it has not been submitted elsewhere for a degree.

Prahlada Rao.

July, 1987

(Dr.D.P. RAO)
Professor
Department of Chemical Engineering
Indian Institute of Technology
Kanpur - 208016
INDIA

ACKNOWLEDGEMENTS

I wish to take this opportunity to express my indebtedness to Professor D.P. Rao for his invaluable guidance. But for his help and encouragement, it would not have been possible to complete this work.

I am very much grateful to Mr. Pankaj Arora who took a lot of trouble and helped during writing this thesis.

I am thankful to all my friends particularly Ashim Dutta, Ashwin Desai and Thomas A. Edison for their assistance in some form or other.

Finally, my gratitude to U.S. Mishra for the efforts he put in typing this material neatly and fast.

-Manoj Kumar Jain

ABSTRACT

An efficient algorithm has been presented for solving the multicomponent, multistage separation process problems involving single columns, columns with pump-around or bypass, or a system of interlinked columns. In this algorithm which is an extension of the Naphtali-Sandholm method, the sparsity and the structure of the submatrices in the Jacobian are exploited, while performing the matrix multiplications and inversions. The operation count was performed for the various matrix multiplications and inversions involved in the algorithm, and using a variety of test-problems it has been shown that the sparsity exploitation results in a significant reduction in the computational and storage requirements. The saving in the computations improves with the increase in the number of components. An efficient approach has been proposed to solve problems with intermediate tray specifications, in which the tridiagonal band structure of the Jacobian is preserved, and has been shown to be more advantageous in both the computational and storage aspects, than the one proposed by Hofeling and Seader.

CONTENTS

CHAPTER 1

INTRODUCTION

The separation processes like distillation, absorption and extraction etc. are widely used in the chemical industries to fractionate the multicomponent mixtures. Complex systems of interlinked columns have been shown to frequently achieve more economic and effective separation of multicomponent mixtures than the conventional single columns and their sequential arrangements. In view of the increasing energy and raw material cost, a continuous evaluation of the performance is required for economic operation as well as to maintain the quality of the product. This requires simulation of multicomponent interlinked columns.

The Naphtali-Sandholm method [1] is widely used for the simulation of single columns due to its good convergence characteristics. For the simulation of interlinked columns a simultaneous rather than a sequential approach is preferred. That is, rather than repeatedly solving a sequence of single column problems until the solution to the entire system is obtained, it is preferred to solve for all the columns simultaneously.

In one version of this approach, the methods like capital-theta method of Holland [10], the equations describing each column are considered separately but all these single column problems are converged simultaneously rather than repeatedly one at a time. Whereas the more powerful approach is to consider the equations describing all the separators simultaneously, and solve by the Newton-Raphson or a similar method with good convergence characteristics. Following this approach several methods have been developed by Hofeling and Seader [2], Kubicek [7,12], Stadtherr [5], Browne [13] etc. While these methods differ in their formulation of equations and their selection of independent variables, all rely on reordering some or all of the linearized equations to produce an almost-band or almost-block-tridiagonal coefficient matrix. Whereas these methods differ in their approach to obtain the desired matrix form and their method of solving the reordered linear system.

The Naphtali-Sandholm method which was originally formulated to solve the single column problems has been extended [2], to handle the system of interlinked columns, and the columns with pump-around or bypass, while retaining the technique of total linearization and simultaneous solution of all the equations in the system by the Newton-Raphson method.

Though this method offers a good convergence characteristics, the large computational and storage requirements are the impediments.

When the Newton-Raphson technique is applied to solve the equations in the Naphtali-Sandholm method, the resulting Jacobian matrix has a tridiagonal-band or almost-tridiagonal-band structure with a few off-tridiagonal elements. The submatrices in the Jacobian are sparse and have a definite structure.

It appears from the literature that the sparsity of the submatrices has not been exploited. The objective of this work is to present an efficient algorithm, which takes the advantage of the sparsity and the structure of the submatrices in the Jacobian, to solve the separation process problems.

In Chapter 2, first the method of formulation and solution for single column problems has been presented and then extended to the case of interlinked columns. Chapter 3 deals with the exploitation of sparsity in the various matrix multiplications and inversions. In Chapter 4, an efficient approach has been proposed to solve the problems with the intermediate tray specifications, and a variety of specifications for condensers, reboilers and intermediate trays have been presented. Chapter 5 deals with the details of implementation of separation process problems on the computer. The results of various test-problems and a brief discussion have been presented in Chapter 6 and the conclusions in Chapter 7.

CHAPTER 2

MODELLING OF MULTICOMPONENT MULTISTAGE COLUMNS

In this chapter, first the modelling of a single multicomponent multistage separation column is presented, which is later extended to a system of interlinked columns. In both the cases, the model proposed by Naphtali and Sandholm [1] , is followed, except for a minor reordering of variables and discrepancy function in the formulation of the Jacobian-matrix, the reason for which it is done is explained later.

2.1 Generalized Tray Model for a Single Column

A model for a general tray j, with streams to and from the neighbouring trays j-1 and j+1, and with side streams is shown in Figure 1. The discrepancy functions formulated using mass balances, equilibrium relations and energy balance are as under:

Component Mass Balances:

$$M_{j,i} = l_{j-1,i} + v_{j+1,i} - (1+s_j)l_{j,i} - (1+S_j)v_{j,i} + f_{j,i}$$

$$\text{for } 1 \leq i \leq C \text{ and } 1 \leq j \leq N \qquad (2.1)$$

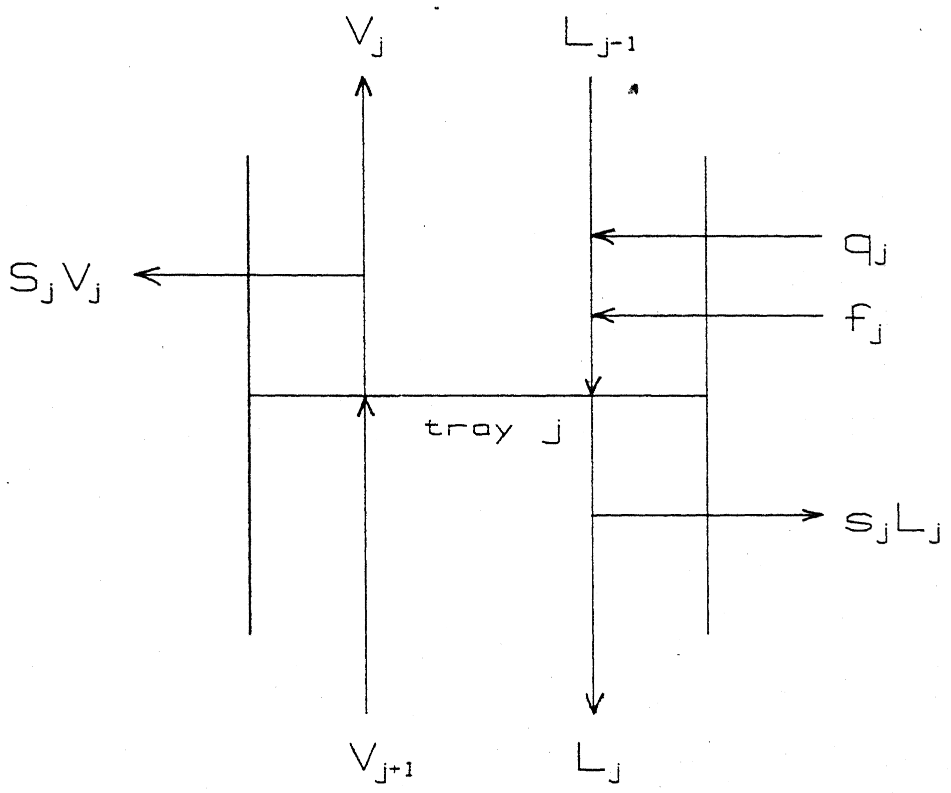


FIG 1 A GENERAL TRAY IN A SINGLE COLUMN

Equilibrium Relationships:

$$Q_{j,i} = \frac{\eta_j K_{j,i}}{L_j} l_{j,i} - \frac{v_{j,i}}{V_j} + \frac{(1 - \eta_j)}{V_{j+1}} v_{j+1,i} \tag{2.2}$$

for $1 \leq i \leq C$ and $1 \leq j \leq N$

where

$$\eta_j = \frac{y_{j,i} - y_{j+1,i}}{K_{j,i}\, x_{j,i} - y_{j+1,i}} \tag{2.3}$$

Energy Balance:

$$E_j = \sum_{i=1}^{c} h_{j-1,i}\, l_{j-1,i} + \sum_{i=1}^{c} H_{j+1,i}\, v_{j+1,i} - (1+s_j) \sum_{i=1}^{c} h_{j,i}\, l_{j,i}$$

$$- (1+ S_j) \sum_{i=1}^{c} H_{j,i}\, v_{j,i} + \sum_{i=1}^{c} h_{F_{j,i}}\, f_{j,i} + q_j \tag{2.4}$$

for $1 \leq j \leq N$

where, $l_{j,i}$ and $v_{j,i}$ are the molar liquid and vapor flow rates leaving the tray j after the liquid and vapor side streams $s_j l_{j,i}$ and $S_j v_{j,i}$ have been drawn from stage j.

Thus there are (2c+1) equations (discrepancy functions), viz. $M_{j,i}$, $Q_{j,i}$ and E_j, and (2c+1) variables, viz. $l_{j,i}$, $v_{j,i}$ and T_j for each tray and hence a total of N(2c+1) of each for the whole column. The set of these N(2c+1) discrepancy functions and variables may be written in a compact form as follows:

$$\bar{F}(\bar{X}) = \bar{0} \tag{2.5}$$

where,

$$\bar{F} = (\bar{F}_1, \bar{F}_2, \ldots, \bar{F}_j, \ldots, \bar{F}_N)^T \tag{2.6}$$

$$\bar{F}_j = (M_{j,1}, M_{j,2}, \ldots, M_{j,c}, Q_{j,1}, \ldots, Q_{j,c}, E_j)^T \tag{2.7}$$

for $1 \leq j \leq N$

and

$$\bar{X} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_j, \ldots, \bar{x}_N)^T \tag{2.8}$$

$$\bar{x}_j = (l_{j,1}, l_{j,2}, \ldots, l_{j,c}, v_{j,1}, v_{j,2}, \ldots, v_{j,c}, T_j)^T \tag{2.9}$$

for $1 \leq j \leq N$

It may be noted here that:

(a) The ordering of the discrepancy functions and variables is a bit different from the one proposed by Naphtali and Sandholm [1] . This reordering results in the suitable structure of submatrices for the purpose of sparsity exploitation, which is explained later.

(b) A minor change has been made in the discrepancy function $Q_{j,i}$, which was proposed by Naphatali and Sandholm as:

$$Q_{j,i} = \frac{\eta_j K_{j,i} V_j}{L_j} l_{j,i} - v_{j,i} + \frac{(1-\eta_j)V_j}{V_{j+1}} v_{j+1,i} \tag{2.10}$$

The discrepancy function (2.2) is obtained by dividing the original equation (2.10) by V_j. This minor change has improved the convergence characteristics.

Employing the Newton-Raphson method to solve the system of equations (2.5) simultaneously we obtain:

$$\Delta \bar{x}_{m+1} = -\left(\frac{d\bar{F}}{d\bar{X}}\right)^{-1}_{m} \bar{\bar{F}}_m \qquad (2.11)$$

$$= - \bar{\bar{J}}^{-1}_m \bar{\bar{F}}_m \qquad (2.12)$$

where, $\bar{J}$ is the Jacobian matrix and m is the iteration number, and,

$$\bar{x}_{m+1} = \bar{x}_m + \Delta \bar{x}_{m+1} \qquad (2.13)$$

The structure of the Jacobian matrix is:

$$\begin{bmatrix}
\bar{\bar{B}}_1 & \bar{\bar{C}}_1 & & & & & & & \\
\bar{\bar{A}}_2 & \bar{\bar{B}}_2 & \bar{\bar{C}}_2 & & & & & & \\
 & \bar{\bar{A}}_3 & \bar{\bar{B}}_3 & \bar{\bar{C}}_3 & & & & & \\
 & & \cdot & \cdot & \cdot & & & & \\
 & & & \cdot & \cdot & \cdot & & & \\
 & & & & \bar{\bar{A}}_j & \bar{\bar{B}}_j & \bar{\bar{C}}_j & & \\
 & & & & & \cdot & \cdot & \cdot & \\
 & & & & & & \cdot & \cdot & \cdot \\
 & & & & & & & \bar{\bar{A}}_N & \bar{\bar{B}}_N
\end{bmatrix} \qquad (2.14)$$

where,

$$\bar{\bar{A}}_j = \frac{\partial \bar{F}_j}{\partial \bar{x}_{j-1}} , \quad \bar{\bar{B}}_j = \frac{\partial \bar{F}_j}{\partial \bar{x}_j} , \quad \bar{\bar{C}}_j = \frac{\partial \bar{F}_j}{\partial \bar{x}_{j+1}} \qquad (2.15)$$

The submatrices $\bar{\bar{A}}_j$, $\bar{\bar{B}}_j$ and $\bar{\bar{C}}_j$ have the following elements:

$$\bar{\bar{A}}_j = \left[\begin{array}{ccccccc}
\frac{\partial M_{j,1}}{\partial l_{j-1,1}} & \cdots & \frac{\partial M_{j,1}}{\partial l_{j-1,c}} & \frac{\partial M_{j,1}}{\partial v_{j-1,1}} & \cdots & \frac{\partial M_{j,1}}{\partial v_{j-1,c}} & \frac{\partial M_{j,1}}{\partial T_{j-1}} \\
\vdots & & \vdots & \vdots & & \vdots & \vdots \\
\frac{\partial M_{j,c}}{\partial l_{j-1,1}} & \cdots & \frac{\partial M_{j,c}}{\partial l_{j-1,c}} & \frac{\partial M_{j,c}}{\partial v_{j-1,1}} & \cdots & \frac{\partial M_{j,c}}{\partial v_{j-1,c}} & \frac{\partial M_{j,c}}{\partial T_{j-1}} \\
\frac{Q_{j,1}}{l_{j-1,1}} & \cdots & \frac{\partial Q_{j,1}}{\partial l_{j-1,c}} & \frac{\partial Q_{j,1}}{\partial v_{j-1,1}} & \cdots & \frac{\partial Q_{j,1}}{\partial v_{j-1,c}} & \frac{\partial Q_{j,1}}{\partial T_{j-1}} \\
\vdots & & \vdots & \vdots & & \vdots & \vdots \\
\frac{\partial Q_{j,c}}{\partial l_{j-1,1}} & \cdots & \frac{\partial Q_{j,c}}{\partial l_{j-1,c}} & \frac{\partial Q_{j,c}}{\partial v_{j-1,1}} & \cdots & \frac{\partial Q_{j,c}}{\partial v_{j-1,c}} & \frac{\partial Q_{j,c}}{\partial T_{j-1}} \\
\frac{\partial E_j}{\partial l_{j-1,1}} & & \frac{\partial E_j}{\partial l_{j-1,c}} & \frac{\partial E_j}{\partial v_{j-1,1}} & \cdots & \frac{\partial E_j}{\partial v_{j-1,c}} & \frac{\partial E_j}{\partial T_{j-1}}
\end{array}\right] \qquad (2.16)$$

$$\bar{B}_j = \begin{bmatrix} \frac{\partial M_{j,1}}{\partial l_{j,1}} & \cdots & \frac{\partial M_{j,1}}{\partial l_{j,c}} & \frac{\partial M_{j,1}}{\partial v_{j,1}} & \cdots & \frac{\partial M_{j,1}}{\partial v_{j,c}} & \frac{\partial M_{j,1}}{\partial T_j} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \frac{\partial M_{j,c}}{\partial l_{j,1}} & \cdots & \frac{\partial M_{j,c}}{\partial l_{j,c}} & \frac{\partial M_{j,c}}{\partial v_{j,1}} & \cdots & \frac{\partial M_{j,c}}{\partial v_{j,c}} & \frac{\partial M_{j,c}}{\partial T_j} \\ \frac{\partial Q_{j,1}}{\partial l_{j,1}} & \cdots & \frac{\partial Q_{j,1}}{\partial l_{j,c}} & \frac{\partial Q_{j,1}}{\partial v_{j,1}} & \cdots & \frac{\partial Q_{j,1}}{\partial v_{j,c}} & \frac{\partial Q_{j,1}}{\partial T_j} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \frac{\partial Q_{j,c}}{\partial l_{j,1}} & \cdots & \frac{\partial Q_{j,c}}{\partial l_{j,c}} & \frac{\partial Q_{j,c}}{\partial v_{j,1}} & \cdots & \frac{\partial Q_{j,c}}{\partial v_{j,c}} & \frac{\partial Q_{j,c}}{\partial T_j} \\ \frac{\partial E_j}{\partial l_{j,1}} & \cdots & \frac{\partial E_j}{\partial l_{j,c}} & \frac{\partial E_j}{\partial v_{j,1}} & \cdots & \frac{\partial E_j}{\partial v_{j,c}} & \frac{\partial E_j}{\partial T_j} \end{bmatrix} \qquad (2.17)$$

$$\bar{C}_j = \begin{bmatrix} \frac{\partial M_{j,1}}{\partial l_{j+1,1}} & \cdots & \frac{\partial M_{j,1}}{\partial l_{j+1,c}} & \frac{\partial M_{j,1}}{\partial v_{j+1,1}} & \cdots & \frac{\partial M_{j,1}}{\partial v_{j+1,c}} & \frac{\partial M_{j,1}}{\partial T_{j+1}} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \frac{\partial M_{j,c}}{\partial l_{j+1,1}} & \cdots & \frac{\partial M_{j,c}}{\partial l_{j+1,c}} & \frac{\partial M_{j,c}}{\partial v_{j+1,1}} & \cdots & \frac{\partial M_{j,c}}{\partial v_{j+1,c}} & \frac{\partial M_{j,c}}{\partial T_{j+1}} \\ \frac{\partial Q_{j,1}}{\partial l_{j+1,1}} & \cdots & \frac{\partial Q_{j,1}}{\partial l_{j+1,c}} & \frac{\partial Q_{j,1}}{\partial v_{j+1,1}} & \cdots & \frac{\partial Q_{j,1}}{\partial v_{j+1,c}} & \frac{\partial Q_{j,1}}{\partial T_{j+1}} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \frac{\partial Q_{j,c}}{\partial l_{j+1,1}} & \cdots & \frac{\partial Q_{j,c}}{\partial l_{j+1,c}} & \frac{\partial Q_{j,c}}{\partial v_{j+1,1}} & \cdots & \frac{\partial Q_{j,c}}{\partial v_{j+1,c}} & \frac{\partial Q_{j,c}}{\partial T_{j+1}} \\ \frac{\partial E_j}{\partial l_{j+1,1}} & \cdots & \frac{\partial E_j}{\partial l_{j+1,c}} & \frac{\partial E_j}{\partial v_{j+1,1}} & \cdots & \frac{\partial E_j}{\partial v_{j+1,c}} & \frac{\partial E_j}{\partial T_{j+1}} \end{bmatrix} \qquad (2.18)$$

In compact form, the structures of $\bar{\bar{A}}$, $\bar{\bar{B}}$ and $\bar{\bar{C}}$ submatrices may be written as

$$\bar{\bar{A}}_j = \begin{bmatrix} \bar{\bar{I}}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{\bar{O}}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \qquad (2.19)$$

$$\bar{\bar{B}}_j = \begin{bmatrix} -(1+s_j)\bar{\bar{I}}_c & -(1+S_j)\bar{\bar{I}}_c & \bar{O}_c \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix} \qquad (2.20)$$

$$\bar{\bar{C}}_j = \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{I}}_c & \bar{O}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{O}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \qquad (2.21)$$

where,

$\bar{\bar{I}}_c$ is an identity matrix of order cxc

$\bar{\bar{O}}_c$ is a null matrix of order cxc

$\bar{O}_c$ is either a row or a column null vector (whose meaning is clear from its position) of order c

$\bar{\bar{X}}_c$ is a matrix with non-zero elements, of order cxc

$\bar{X}_c$ is either a row or a column vector with non-zero elements, of order c

X is a non-zero element.

It may be noted that the Jacobian matrix in (2.14) has a tridiagonal-band-structure and is highly sparse. This is due to the fact that only those derivatives are non-zero which are obtained by differentiating the discrepancy functions of any tray j, with respect to the variables of the same tray j, or the variables of the neighbouring trays j-1 or j+1.

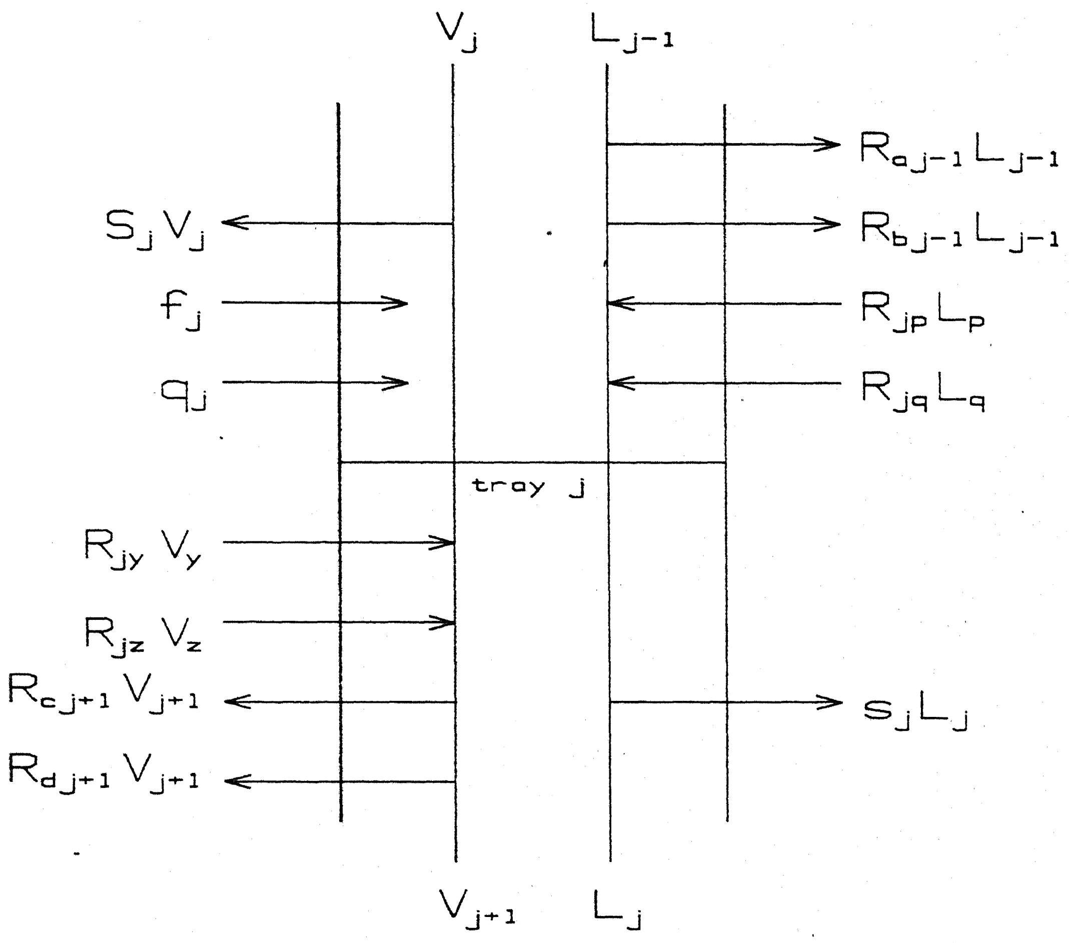
Further, the submatrices $\bar{\bar{A}}$ and $\bar{\bar{C}}$ in the Jacobian $\bar{\bar{J}}$ are also sparse which may be noted in the structures presented in (2.19) and (2.21).

2.2 Interlinked Columns:

The Naphtali-Sandholm method which was originally formulated to solve the single column problems, can be extended to handle a system of interlinked columns, and the columns with pump-around or bypass.

Generalized Tray Model:

A model for a general tray j, in a system of interlinked columns or in the columns with bypass or pump-around streams is presented in Figure 2. In addition to the streams to and from the neighbouring trays and side-streams, this model



$$r_{L_{j-1}} = 1-(R_{aj-1}+R_{bj-1}+\dots\quad)$$

$$r_{V_{j+1}} = 1-(R_{cj+1}+R_{dj+1}+\dots\quad)$$

FIG 2 A GENERAL TRAY IN A SYSTEM OF INTERLINKED COLUMNS

includes the interlinked-streams which link the tray j to trays which are not its immediate neighbours in the system-model.

Let $R_{jp} L_p$, $R_{jq} L_q$, $R_{jy} V_y$, $R_{jz} V_z$ ... be the streams which are leaving the tray p, q, y, z ..., and are fed to the stage j, where R_{kj} represents the fraction of stream which is leaving the tray k and fed to tray j, and

R_{kj} = 0 if there is no stream from tray k to tray j
= 1 if the whole stream from k is fed to tray j
= a a value between 0 and 1 if a part of the stream leaving the tray k is fed to tray j.

The sum of all liquid and vapor interlinked streams leaving the tray j is represented by $(1 - r_{L_{j-1}})L_{j-1}$ and $(1 - r_{V_{j+1}})V_{j+1}$, respectively.

The discrepancy functions formulated based on the component material balances, equilibrium relationships and enthalpy balance are:

Component Mass Balances:

$$M_{j,i} = r_{L_{j-1}} l_{j-1,i} + r_{V_{j+1}} v_{j+1,i} - (1+s_j) l_{j,i} - (1+S_j) v_{j,i}$$
$$+ R_{jp} l_{p,i} + R_{jq} l_{q,i} + \dots + R_{jy} v_{y,i} + R_{jz} v_{z,i} + \dots$$
$$+ f_{j,i} \qquad \text{for } 1 \leq i \leq C \text{ and } 1 \leq j \leq N \qquad (2.22)$$

Equilibrium Relationships:

$$Q_{j,i} = \frac{\eta_j K_{j,i}}{L_j} l_{j,i} - \frac{v_{j,i}}{V_j} + \frac{(1-\eta_j) r_{V_{j+1}} v_{j+1,i}}{r_{V_{j+1}} V_{j+1} + R_{jy} V_y + R_{jz} V_z} + \frac{(1-\eta_j) R_{jy} v_{y,i}}{r_{V_{j+1}} V_{j+1} + R_{jy} V_y + R_{jz} V_z}$$

$$+ \frac{(1-\eta_j) R_{jz} v_{z,i}}{r_{V_{j+1}} V_{j+1} + R_{jy} V_y + R_{jz} V_z} + \ldots \qquad (2.23)$$

for $1 \leq i \leq C$ and $1 \leq j \leq N$

Enthalpy Balance:

$$E_j = r_{L_{j-1}} \sum_{i=1}^{c} h_{j-1,i} l_{j-1,i} + r_{V_{j+1}} \sum_{i=1}^{c} H_{j+1,i} v_{j+1,i}$$

$$-(1+s_j) \sum_{i=1}^{c} h_{j,i} l_{j,i} - (1+S_j) \sum_{i=1}^{c} H_{j,i} v_{j,i}$$

$$+ R_{jp} \sum_{i=1}^{c} h_{p,i} l_{p,i} + R_{jq} \sum_{i=1}^{c} h_{q,i} l_{q,i} + \ldots.$$

$$+ R_{jy} \sum_{i=1}^{c} H_{y,i} v_{y,i} + R_{jz} \sum_{i=1}^{c} H_{z,i} v_{z,i} + \ldots.$$

$$+ \sum_{i=1}^{c} h_{F_{j,i}} f_{j,i} + q_j \quad \text{for } 1 \leq j \leq N \qquad (2.24)$$

The choice of variable remains the same as it was for a single column viz. $l_{j,i}$, $v_{j,i}$ and T_j, in that order. Thus, we have N(2c+1) equations and N(2c+1) variables which are to be solved simultaneously using the Newton-Raphson method.

It may be noted here that the discrepancy functions of the tray j may also have the variables of the trays, which are not the immediate neighbours of the tray j, due to the presence of interlinked-streams, bypass or pump-arounds. Therefore, the Jacobian matrix may have some off-tridiagonal blocks in addition to the tridiagonal-band. The position and structure of these off-tridiagonal blocks depend upon the arrangement of the columns in the system model (Hildalgo and Seader [4]), position of the interlinked trays, and the direction and type (liquid or vapor) of the interlinked streams.

The single stream between two trays (not neighbours) may be classified into four basic types:

(i) liquid stream from an upper tray to a lower tray
(ii) liquid stream from a lower tray to an upper tray
(iii) vapor stream from a lower tray to an upper tray
(iv) vapor stream from an upper tray to a lower tray.

The reciprocal streams (in which a stream from one stage to another stage, is matched by a stream of the other phase flowing in the opposite direction between the same two stages), may be handled as a combination of these four types of streams.

Consider an arrangement of interlinked-columns in which j and q are the two trays such that j is above q and $q-j \geq 2$, and a stream linking the two. The presence of this stream will cause an off-diagonal block to occur in the Jacobian matrix, and the position and structure of which may be determined considering the four types discussed below:

Type 1: A liquid stream $R_{qj}L_j$ from the tray j to tray q. (Liquid flowing downward)

$$\bar{\bar{J}} = \begin{bmatrix} B_1 & C_1 & & & & & & & \\ A_2 & B_2 & C_2 & & & & & & \\ & & \cdot & \cdot & \cdot & & & & \\ & & & A_j & B_j & C_j & & & \\ & & & & \vdots & \cdot & \cdot & \cdot & \\ & & & & A_{q,j} & \cdots A_q & B_q & C_q & \\ & & & & & & \cdot & \cdot & \cdot \\ & & & & & & & A_N & B_N \end{bmatrix}$$

where

$$\bar{\bar{A}}_{q,j} = \begin{bmatrix} R_{qj}\,\bar{\bar{I}}_c & \bar{\bar{0}}_c & \bar{0}_c \\ \bar{\bar{0}}_c & \bar{\bar{0}}_c & \bar{0}_c \\ \bar{X}_c & \bar{0}_c & X \end{bmatrix}$$

Structure: similar to the tri-diagonal A blocks

Position : below the tridiagonal

Type 2: A liquid stream $R_{jq}L_q$ from the tray q to tray j (Liquid flowing upward)

$$\bar{\bar{J}} = \begin{bmatrix}
B_1 & C_1 & & & & & & & & & \\
A_2 & B_2 & C_2 & & & & & & & & \\
 & \ddots & \ddots & \ddots & \ddots & & & & & & \\
 & & & A_j & B_j & C_j & \cdots & C_{j,q} & & & \\
 & & & & \ddots & \ddots & \ddots & \vdots & & & \\
 & & & & & & A_q & B_q & C_q & & \\
 & & & & & & & & \ddots & \ddots & \ddots \\
 & & & & & & & & & A_N & B_N
\end{bmatrix}$$

where,

$$\bar{\bar{C}}_{j,q} = \begin{bmatrix}
R_{j,q}\bar{\bar{I}}_c & \bar{\bar{O}}_c & \bar{O}_c \\
\bar{\bar{O}}_c & \bar{\bar{O}}_c & \bar{O}_c \\
\bar{X}_c & \bar{O}_c & X
\end{bmatrix}$$

Structure: similar to the tridiagonal A-blocks

Position: above the tri-diagonal

Type 3: A vapor stream $R_{jq}V_q$ from tray q to tray j (Vapor flowing upward)

$$\bar{J} = \begin{bmatrix}
B_1 & C_1 & & & & & & & & & \\
A_2 & B_2 & C_2 & & & & & & & & \\
 & \ddots & \ddots & \ddots & & & & & & & \\
 & & & A_j & B_j & C_j & \cdots & C_{j,q} & & & \\
 & & & & \ddots & \ddots & \ddots & \vdots & & & \\
 & & & & & & A_q & B_q & C_q & & \\
 & & & & & & & & \ddots & \ddots & \ddots \\
 & & & & & & & & & A_N & B_N
\end{bmatrix}$$

and

$$\bar{C}_{j,q} = \begin{bmatrix} \bar{\bar{0}}_c & R_{jq}\,\bar{\bar{I}}_c & \bar{0}_c \\ \bar{\bar{0}}_c & \bar{\bar{X}}_c & \bar{0}_c \\ \bar{0}_c & \bar{X}_c & X \end{bmatrix}$$

Structure: similar to tridiagonal C-blocks

Position : above the diagonal

Type 4: A vapor stream $R_{qj}\ V_j$ from tray j to tray q. (Vapor flowing downward)

$$J = \begin{bmatrix} B_1 & C_1 & & & & & & & \\ A_2 & B_2 & C_2 & & & & & & \\ & \cdot & \cdot & \cdot & & & & & \\ & & A_j & B_j & C_j & & & & \\ & & & \cdot & \cdot\quad\cdot & \cdot & & & \\ & & & \vdots & & & & & \\ & & & C_{q,j} & \cdots\cdots & A_q & B_q & C_q & \\ & & & & & & \cdot & \cdot & \cdot \\ & & & & & & & A_N & B_N \end{bmatrix}$$

and

$$C_{q,j} = \begin{bmatrix} \bar{\bar{0}}_c & R_{qj}\bar{\bar{I}}_c & \bar{0}_c \\ \bar{\bar{0}}_c & \bar{\bar{X}}_c & \bar{0}_c \\ \bar{0}_c & \bar{X}_c & X \end{bmatrix}$$

Structure: similar to tri-diagonal c-blocks

Position: below the diagonal

The reciprocal streams in an arrangement of columns cause more than one off-tridiagonal blocks to occurs in the Jacobian, one offdiagonal element per single stream. The reciprocal streams may be handled by breaking them up into single streams and the position and structure all off-tridiagonal blocks can be determined by identifying its type as discussed earlier.

Consider a reciprocal stream between two trays, a liquid stream $R_{jq}L_q$, flowing from tray q to tray j and , a vapor stream $R_{qj}V_j$, flowing from tray j to tray q. The liquid and vapor streams may be identified as type 2 and type 4 individually, and thus the structure of the Jacobian is:

$$\bar{\bar{J}} = \begin{bmatrix} B_1 & C_1 & & & & & & & \\ A_2 & B_2 & C_2 & & & & & & \\ & \vdots & \vdots & \vdots & & & & & \\ & & A_j & B_j & C_j & \cdots\cdots & C_{j,q} & & \\ & & & \vdots & \vdots & \vdots & \vdots & & \\ & & & A_{q,j} & \cdots & A_q & \cdots\cdots B_q & C_q & \\ & & & & & & & \cdots\cdots & \cdots \\ & & & & & & & A_N & B_N \end{bmatrix}$$

where,

$$A_{q,j} = \begin{bmatrix} \bar{\bar{O}}_c & R_{qj}\bar{\bar{I}}_c & \bar{O}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{O}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix}$$

Structure: similar to c-blocks
Position : below the diagonal
(corresponds to the vapor stream)

and

$$C_{j,q} = \begin{bmatrix} R_{j,q}\bar{\bar{I}}_c & \bar{\bar{0}}_c & \bar{0}_c \\ \bar{\bar{0}}_c & \bar{\bar{0}}_c & \bar{0}_c \\ \bar{X}_c & \bar{0}_c & X \end{bmatrix}$$

Structure: similar to A-blocks

Position : above the diagonal

(Corresponds to the liquid stream)

Thus, the position of an off-diagonal element depends only on the direction of the corresponding stream. In the Jacobian matrix, it is positioned in the row corresponding to the tray into which the stream enters, and the column corresponds to the tray from which it exits. Therefore, if the stream flows upward its position would be above the tridiagonal, and below the tridiagonal if the stream flows downwards, in a given arrangement of columns.

The structure of the off-diagonal elements may be determined only by knowing whether it is a liquid or a vapor stream. An offdiagonal element corresponding to a vapor stream has a structure similar to that of a tridiagonal C-block, and similar to that of a tridiagonal A-block if it is a liquid stream.

The detailed calculations of the elements of the tridiagonal and offdiagonal elements have been presented in the Appendix A.

2.3 Modelling of Condensers and Reboilers

Although the condensers and reboilers are treated like trays, the generalized tray model, discussed in

Section 2.1, cannot directly be applied since in these cases a few streams are missing and some additional specifications are required. In a condenser the liquid stream l_{j-1}, and in a reboiler the vapor stream v_{j+1} are not present. Moreover, there may be an additional specification for each condenser and reboiler (These may be specified in various ways, 13 of them are included in the Chapter 4).

We assume that there are no interlinking streams present in the condensers and reboilers, and the heat-duties are specified . (Refer figure 3 for various types of condensers and reboilers).

<u>Partial Condensers:</u>

Component Mass Balances:

$$M_{j,i} = r_{vj+1}\, v_{j+1,i} - (1+S_j)v_{j,i} - (1+s_j)l_{j,i} \qquad 1 \le i \le C \qquad (2.25)$$

Equilibrium Relationships:

$$Q_{j,i} = \frac{\eta_j K_{j,i}}{L_j} l_{j,i} - \frac{v_{j,i}}{V_j} + \frac{(1-\eta_j)}{V_{j+1}} v_{j+1,i} \qquad 1 \le i \le C \qquad (2.26)$$

Enthalpy Balance:

$$E_j = r_{vj+1} \sum_{i=1}^{C} H_{j+1,i}\, v_{j+1,i} - (1+S_j) \sum_{i=1}^{C} H_{j,i}\, v_{j,i} - (1+s_j) \sum_{i=1}^{C} h_{j,i}\, l_{j,i} + Q_c \qquad (2.27)$$

where, Q_c is the condenser-heat-duty.

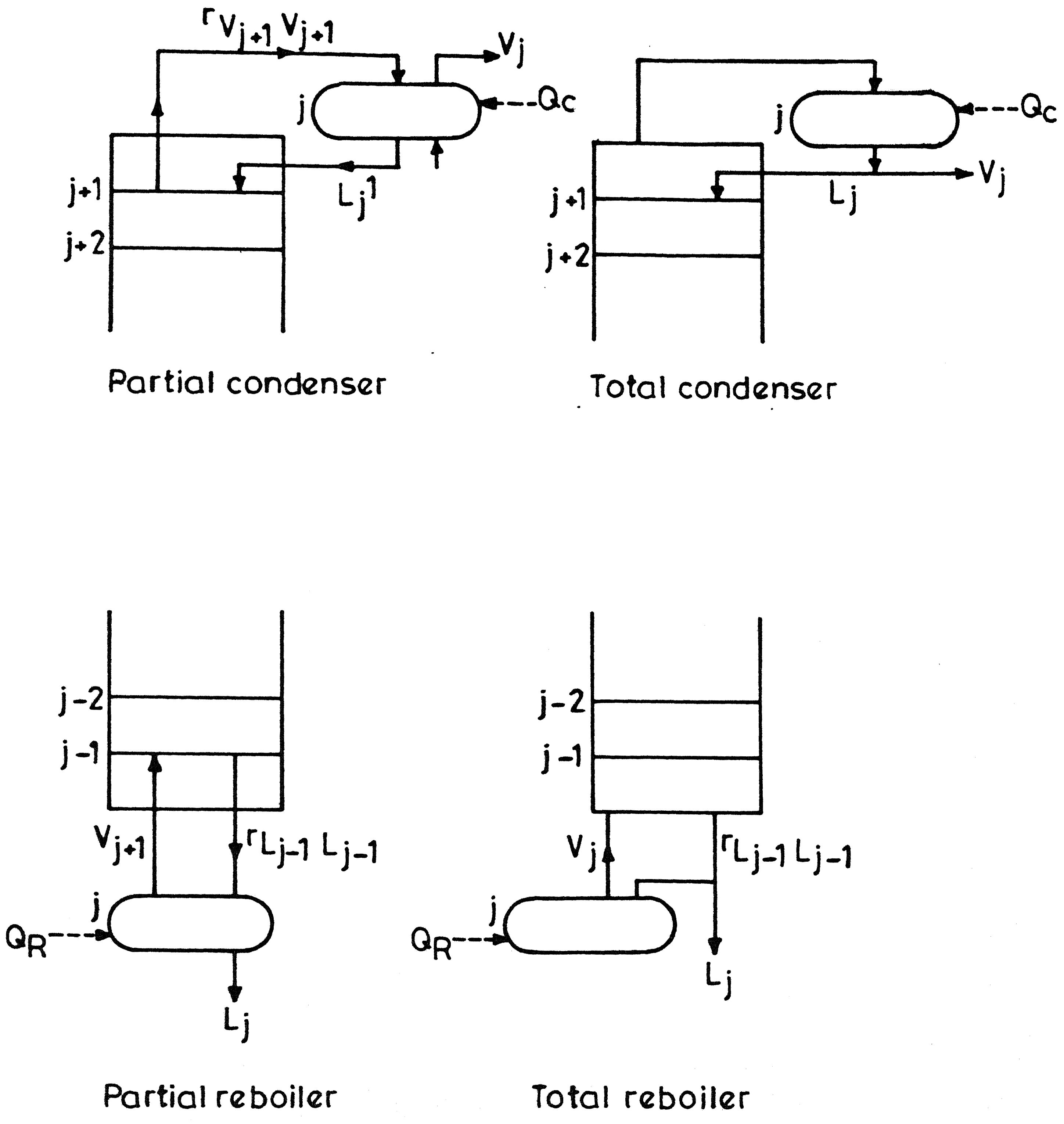

$^{r}V_{j+1}\ V_{j+1}$
V_j
j
Q_c
$j+1$
$j+2$
L_j^1
Partial condenser
j
Q_c
$j+1$
$j+2$
L_j
V_j
Total condenser
$j-2$
$j-1$
V_{j+1}
$^{r}L_{j-1}\ L_{j-1}$
j
Q_R
L_j
Partial reboiler
$j-2$
$j-1$
V_j
$^{r}L_{j-1}\ L_{j-1}$
j
Q_R
L_j
Total reboiler

Fig. 3

Total Condensers:

In the case of total condensers both the streams leaving the condenser are liquid streams, and have the same composition. We use the notation V_j for the distillate (which is liquid).

Component mass balances:

$$M_{j,i} = r_{vj+1}\, v_{j+1,i} - (1 + S_j)v_{j,i} - (1 + s_j)l_{j,i},\quad 1 \leq i \leq C \tag{2.28}$$

Equilibrium relationships: (Since there is no equilibrium, these are replaced by $x_{j,i} = y_{j,i}$ equations)

$$Q_{j,i} = L_j\, v_{j,i} - V_j\, l_{j,i},\quad 1 \leq i \leq C \tag{2.29}$$

Enthalpy Balance:

$$E_j = r_{vj+1} \sum_{i=1}^{C} H_{j+1,i}\, v_{j+1,i} - \sum_{i=1}^{C} h_{j,i}[(1+S_j)v_{j,i} + (1+s_j)l_{j,i}] + Q_c \tag{2.30}$$

Partial Reboilers:

Component mass balances:

$$M_{j,i} = r_{L_{j-1}}\, l_{j-1,i} - (1+s_j)l_{j,i} - (1+S_j)v_{j,i},\quad 1 \leq i \leq C \tag{2.31}$$

Equilibrium Relationships:

$$Q_{j,i} = \frac{K_{j,i}}{L_j} l_{j,i} - \frac{v_{j,i}}{V_j}, \quad 1 \leq i \leq C \qquad (2.32)$$

Enthalpy Balance:

$$E_j = r_{Lj-1} \sum_{i=1}^{C} h_{j-1,i}\, l_{j-1} - (1+s_j) \sum_{i=1}^{C} h_{j,i}\, l_{j,i} - - (1+S_j) \sum_{i=1}^{C} H_{j,i}\, v_{j,i} + Q_R \qquad (2.33)$$

where Q_R is the reboiler-heat-duty.

<u>Total Reboiler:</u>

In the case of a total reboiler a part of the L_{j-1} stream is drawn out as the bottom stream L_j, and the remaining is vaporized and fed to the column as V_j. Therefore, the bottom stream L_j is at the same temperature as that of L_{j-1}, and compositions of L_j and V_j streams are the same.

Component mass balances:

$$M_{j,i} = r_{Lj-1}\, l_{j-1,i} - (1+s_j) l_{j,i} - (1+S_j) v_{j,i} \quad 1 \leq i \leq C \qquad (2.34)$$

Equilibrium Relationships: (Since there is no equilibrium, these are replaced by $x_{j,i} = y_{j,i}$ equations)

$$Q_{j,i} = L_j\, v_{j,i} - V_j\, l_{j,i} \quad 1 \leq i \leq C \qquad (2.35)$$

Enthalpy Balance:

$$E_j = \sum_{i=1}^{c} h_{j-1,i} \left[r_{Vj-1} \, l_{j-1,i} - (1+s_j) l_{j,i} \right] - (1+S_j) \sum_{i=1}^{c} H_{j,i} \, v_{j,i} + Q_R \qquad (2.36)$$

There may be various other specifications besides the condenser and reboiler heat-duty specifications, in which case the enthalpy balances discussed above will no longer be applicable. These are therefore replaced by some other suitable equations which take into account the given specifications. A variety of specifications and alternative discrepancy functions to replace E_j are presemted in Chapter 4.

With the replacement of the discrepancy function E_j, its partial derivatives will also change. Therefore the last rows of the submatrices (B & C in case of condensers and A & B for reboilers) must be modified accordingly.

2.4 Method of Solution

In case of a single column problem with no bypassing or pump arounds, the Jacobian has a completely block-tridiagonal structure, and the well known Thomas-alogorithm may be applied to obtain the solution.

<u>Thomas Algorithm:</u>

Forward Substitution:

Step 1: $\bar{\bar{P}}_1 \leftarrow (\bar{\bar{B}}_1)^{-1} \bar{\bar{C}}_1$

Step 2: $\bar{Q}_1 \leftarrow (\bar{\bar{B}}_1)^{-1} \bar{F}_1$

For stages j, from 2 to (N-1)

Step 3: $\bar{\bar{P}}_j \leftarrow (\bar{\bar{B}}_j - \bar{\bar{A}}_j \bar{\bar{P}}_{j-1})^{-1} \bar{\bar{C}}_j$

Step 4: $\bar{Q}_j \leftarrow (\bar{\bar{B}}_j - \bar{\bar{A}}_j \bar{\bar{P}}_{j-1})^{-1} (\bar{F}_j - \bar{\bar{A}}_j \bar{Q}_{j-1})$

For stage N

Step 5: $\bar{Q}_N \leftarrow (\bar{\bar{B}}_N - \bar{\bar{A}}_N \bar{\bar{P}}_{N-1})^{-1} (\bar{F}_N - \bar{\bar{A}}_N \bar{Q}_{N-1})$

Backward Substitution:

Step 6: $\Delta\bar{X}_N \leftarrow \bar{Q}_N$

For stages j, from (N-1) to 1

Step 7: $\Delta\bar{X}_j \leftarrow (\bar{Q}_j - \bar{\bar{P}}_j \, \Delta\bar{X}_{j+1})$

If in a column bypass or pump-arounds are present, or if an arrangement of interlinked column is considered, the Jacobian does not have a strictly block-tridiagonal structure. Instead, a few off-tridiagonal blocks occur in the Jacobian, and consequently, the conventional Thomas algorithm cannot directly be applied.

A modification of the Thomas algorithm was developed by Hofeling and Seader[2], to solve a system in which the Jacobian has a few off-tridiagonal blocks.

Hofeling and Seader [2], have taken a specific problem to illustrate how the off-tridiagonal blocks could be handled by the modified Thomas algorithm.

Seader [6], justified that using the **same** principles of the modified Thomas algorithm, any arrangement of the off-tridiagonal blocks in a Jacobian could be handled.

There are a numerous ways, in which the off-tridiagonal blocks may appear in the Jacobian, varying from problem to problem. Moreover, for a given system of interlinked columns, there may be a large number of different possible arrangement of these off-tridiagonal blocks in the Jacobian, depending upon the ordering of the columns or column units (refer Hildalgo-Seader[4]). Although the ways to handle these blocks, by the modified Thomas algorithm, are the same in principle, it is difficult to develop a step by step generalized algorithm to handle all possible arrangements of the off-tridiagonal blocks in a Jacobian.

Consider the system of interlinked columns, in which there are two absorbers A_1 and A_2 of 10 plates each, and two distillation columns D_1 and D_2 having 15 and 12 plates respectively. These columns are interlinked as shown in the Figure 6. The Jacobian-matrix, for this system, has six off-tridiagonal blocks, as shown in Figure 7.

We employ the modified Thomas algorithm to solve this system. Various steps of the forward substitution and finally of the backward substitution to obtain a correction vector are presented here.

Modified Thomas Algorithm:

Step 1: row 1, $P_1 \leftarrow B_1^{-1} C_1$; $P_{1,20} \leftarrow B_1^{-1} C_{1,20}$; $Q_1 \leftarrow B_1^{-1} F_1$

Step 2: rows j, where $2 \leq j \leq 9$, $B_j \leftarrow (B_j - A_j P_{j-1})^{-1}$;

$P_j \leftarrow B_j C_j$; $P_{j,20} \leftarrow -B_j A_j P_{j-1,20}$; $Q_j \leftarrow B_j(F_j - A_j Q_{j-1})$

Step 3: row 10, $B_{10} \leftarrow (B_{10} - A_{10} P_9)^{-1}$

$P_{10} \leftarrow \bar{\bar{0}}$; $P_{10,20} \leftarrow -B_{10} A_{10} P_{9,20}$; $Q_{10} \leftarrow B_{10}(F_{10} - A_{10} Q_9)$

Step 4: row 11, $P_{11} \leftarrow B_{11}^{-1} C_{11}$; $P_{11,47} \leftarrow B_{11}^{-1} C_{11,47}$;

$P_{11,20} \leftarrow \bar{\bar{0}}$; $Q_{11} \leftarrow B_{11}^{-1} F_{11}$

Step 5: row j, where $12 \leq j \leq 18$, $B_j \leftarrow (B_j - A_j P_{j-1})^{-1}$

$P_j \leftarrow B_j C_j$; $P_{j,20} \leftarrow \bar{\bar{0}}$; $P_{j,47} \leftarrow -B_j A_j P_{j-1,47}$;

$Q_j \leftarrow B_j(F_j - A_j Q_{j-1})$

Step 6: row 19, $B_{19} \leftarrow (B_{19} - A_{19} P_{18})^{-1}$

$P_{19} \leftarrow B_{19}C_{19}$; $P_{19,47} \leftarrow -B_{19}A_{19}P_{18,47}$;

$Q_{19} \leftarrow B_{19}(F_{19} - A_{19}Q_{18})$

Step 7: row 20, set $\beta_1 \leftarrow Q_{10}$; set $\beta_2 \leftarrow P_{10,20}$;

iterate on $\beta_1 \leftarrow Q_j - P_j\beta_1$, from $j = 9$ to $j = 1$;

iterate on $\beta_2 \leftarrow P_{j,20} - P_j\beta_2$, from $j = 9$ to $j = 1$;

$B_{20} \leftarrow (B_{20} - A_{20}P_{19} - A_{20,1}\beta_2)^{-1}$

$P_{20} \leftarrow \bar{0}$; $P_{20,47} \leftarrow -B_{20}A_{20}P_{19,47}$;

$Q_{20} \leftarrow B_{20}(F_{20} - A_{20}Q_{19} - A_{20,1}\beta_1)$

Step 8: row 21, $P_{21} \leftarrow B_{21}^{-1} C_{21}$; $Q_{21} \leftarrow B_{21}^{-1} F_{21}$; $P_{21,47} \leftarrow \bar{0}$

Step 9: row 22, $B_{22} \leftarrow (B_{22} - A_{22}P_{21})^{-1}$;

$P_{22} \leftarrow B_{22}C_{22}$; $P_{22,47} \leftarrow \bar{0}$; $Q_{22} \leftarrow B_{22}(F_{22} - A_{22}Q_{21})$

Step 10: row 23, $B_{23} \leftarrow (B_{23} - A_{23}P_{22})^{-1}$

$P_{23} \leftarrow B_{23}C_{23}$; $P_{23,47} \leftarrow B_{23}A_{23,10}P_{10,20}P_{20,47}$

$Q_{23} \leftarrow B_{23}[F_{23} - A_{23}Q_{22} - A_{23,10}(Q_{10} - P_{10,20}Q_{20})]$

Step 11: rows j, where $24 \leq j \leq 34$; $B_j \leftarrow (B_j - A_jP_{j-1})^{-1}$

$P_j \leftarrow B_jC_j$; $P_{j,47} \leftarrow -B_jA_jP_{j-1,47}$; $Q_j \leftarrow B_j(F_j - A_jQ_{j-1})$

Step 12: row 35, $B_{35} \leftarrow (B_{35} - A_{35}P_{34})^{-1}$

$P_{35} \leftarrow \bar{0}$; $P_{35,47} \leftarrow -B_{35}A_{35}P_{34,47}$; $Q_{35} \leftarrow B_{35}(F_{35} - A_{35}Q_{34})$

Step 13: row 36, $P_{36} \leftarrow B_{36}^{-1} C_{36}$; $P_{36,47} \leftarrow \bar{\bar{0}}$; $Q_{36} \leftarrow B_{36}^{-1} F_{36}$

Step 14: rows j, where $37 \leq j \leq 39$; $B_j \leftarrow (B_j - A_j P_{j-1})^{-1}$

$P_j \leftarrow B_j C_j$; $P_{j,47} \leftarrow \bar{\bar{0}}$; $Q_j \leftarrow B_j (F_j - A_j Q_{j-1})$

Step 15: row 40, $B_{40} \leftarrow (B_{40} - A_{40} P_{39})^{-1}$

reset $\beta_1 \leftarrow Q_{35}$;

iterate on $\beta_1 \leftarrow Q_j - P_j \beta_1$, from $j = 34$ to $j = 21$

reset $\beta_2 \leftarrow P_{35,47}$;

iterate on $\beta_2 \leftarrow P_{j,47} - P_j \beta_2$, from $j = 34$ to $j = 23$

$\beta_2 \leftarrow P_{21} P_{22} \beta_2$

$P_{40} \leftarrow B_{40} C_{40}$; $P_{40,47} \leftarrow -B_{40} A_{40,21} \beta_2$;

$Q_{40} \leftarrow B_{40} (F_{40} - A_{40} Q_{39} - A_{40,21} \beta_1)$

Step 16: row 41, $B_{41} \leftarrow (B_{41} - A_{41} P_{40})^{-1}$;

$P_{41} \leftarrow B_{41} C_{41}$; $P_{41,47} \leftarrow [- B_{41} (A_{41} P_{40,47} + A_{41,20} P_{20,47})]$

$Q_{41} \leftarrow B_{41} (F_{41} - A_{41} Q_{40} - A_{41,20} Q_{20})$

Step 17: rows j, where $42 \leq j \leq 45$, $B_j \leftarrow (B_j - A_j P_{j-1})^{-1}$

$P_j \leftarrow B_j C_j$; $P_{j,47} \leftarrow -B_j A_j P_{j-1,47}$; $Q_j \leftarrow B_j (F_j - A_j Q_{j-1})$

Step 18: row 46, $B_{46} \leftarrow (B_{46} - A_{46} P_{45})^{-1}$

$P_{46} \leftarrow B_{46} (C_{46} - A_{46} P_{45,47})$; $Q_{46} \leftarrow B_{46} (F_{46} - A_{46} Q_{45})$

Step 19: row 47, $Q_{47} \leftarrow (B_{47} - A_{47} P_{46})^{-1} (F_{47} - A_{47} Q_{46})$

Back Substitution:

Step 1: row 47, $X_{47} \leftarrow Q_{47}$

Step 2: row 46, $X_{46} \leftarrow Q_{46} - P_{46}X_{47}$

Step 3: rows j, from j = 45 to j = 40

$$X_j \leftarrow Q_j - P_j X_{j+1} - P_{j,47} X_{47}$$

Step 4: rows j, from j = 39 to j = 36

$$X_j \leftarrow Q_j - P_j X_{j+1}$$

Step 5: row 35, $X_{35} \leftarrow Q_{35} - P_{35,47} X_{47}$

Step 6: rows j, from j = 34 to j = 23

$$X_j \leftarrow Q_j - P_j X_{j+1} - P_{j,47} X_{47}$$

Step 7: rows j, from j = 22 and j = 21

$$X_j \leftarrow Q_j - P_j X_{j+1}$$

Step 8: row 20, $X_{20} \leftarrow Q_{20} - P_{20,47} X_{47}$

Step 9: rows j, from j = 19 to j = 11

$$X_j \leftarrow Q_j - P_j X_{j+1} - P_{j,47} X_{47}$$

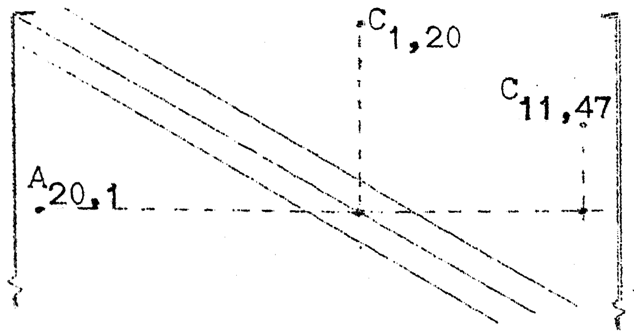
Step 10: row 10, $X_{10} \leftarrow Q_{10} - P_{10,20} X_{20}$

Step 11: rows j, from j = 9 to j =1

$$X_j \leftarrow Q_j - P_j X_{j+1} - P_{j,20} X_{20}$$

It may be noted here that the calculations for eliminating the lower off-tridiagonal blocks is different for the blocks $A_{20,1}$, $A_{23,10}$ and $A_{40,21}$ and $A_{41,20}$. Each case is discussed here separately:

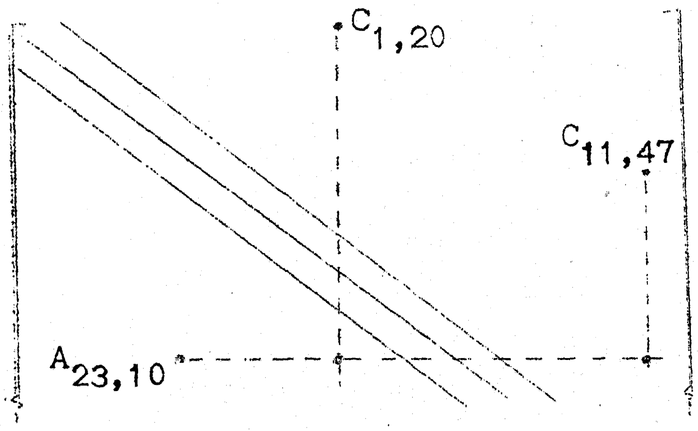
Row 20: Block $A_{20,1}$: The structure of Jacobian between rows 1 and 20 is as under:



Here, the row of $A_{20,1}$ and column of $C_{1,20}$ intersect on the diagonal, whereas those of $A_{20,1}$ and $C_{11,47}$ intersect at a point above the diagonal.

The calculation procedure for this type of situation is included in the step 7. Since the C_{20} block is zero, P_{20} block is also zero.

Row 23: Block $A_{23,10}$: The structure of the Jacobian up to row 23, is as follows:



In this case, the row of $A_{23,10}$ and the column of $C_{1,20}$ intersect at a point below the diagonal, whereas those of $A_{23,10}$ and $C_{11,40}$, above the diagonal.

The calculation procedure for this kind of situation is different from the previous case. Refer step 10. (It may be pointed out here that since C_{10} and C_{20} are zero therefore the blocks of rows between 10 and 20 and those of rows between 20 to 23 do not appear in the calculations). Please refer Figure 7.

If the blocks C_{10} and C_{20} were not zeros, then the step 10 would have been:

```
row 23, set $\beta_1 \leftarrow Q_{19}$,
iterate on $\beta_1 \leftarrow Q_j - P_j\beta_1$, from j = 18 to j = 10;
        set $\beta_2 \leftarrow P_{19,47}$,
iterate on $\beta_2 \leftarrow P_{j,47} - P_j\beta_2$, from j = 18 to j = 11;
       $\beta_2 \leftarrow P_{10}\beta_2$;
       set $\beta_3 \leftarrow P_{19,20}$
iterate on $\beta_3 \leftarrow P_{j,20} - P_j\beta_2$, from j = 18to j = 11;
$\beta_3 \leftarrow P_{10}\beta_3$;
       set $\alpha_1 \leftarrow Q_{22}$
iterate on $\alpha_1 \leftarrow Q_j - P_j\alpha_1$, from j = 21 to j = 20
       set $\alpha_2 \leftarrow P_{22,47}$
iterate on $\alpha_2 \leftarrow P_{j,47} - P_j\alpha_2$, from j = 21 to j = 20
       set $\alpha_3 \leftarrow P_{22}$
```

iterate on $\alpha_3 \leftarrow P_j\alpha_3$, from j = 21 to j = 20

$B_{23} \leftarrow (B_{23} - A_{23}P_{22} - A_{23,10}\beta_3\alpha_3)^{-1}$

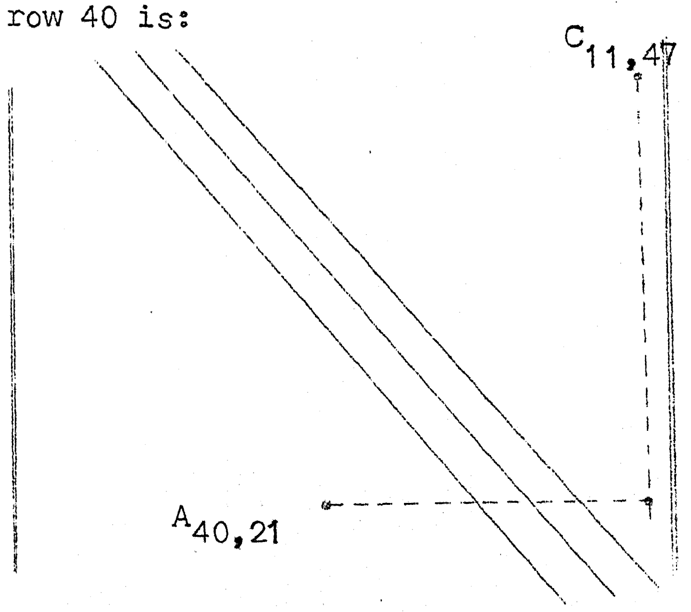
$P_{23} \leftarrow B_{23}C_{23}$;

$P_{23,47} \leftarrow B_{23}[A_{23,10}(\beta_2 - \beta_3\alpha_2) - A_{23}P_{22,47}]$

$Q_{23} \leftarrow B_{23}[F_{23} - A_{23}Q_{22} - A_{23,10}(\beta_1 + \beta_3\gamma_1)]$

Thus, the calculation procedure for this type of situation, becomes some what more complicated than the one discussed earlier.

<u>Row 40</u>: Block $A_{40,21}$: The structure of the Jacobian upto row 40 is:



Here, the row of $A_{40,21}$ and the column of $C_{11,47}$ intersect at a point above the diagonal and there are no other off-tridiagonal blocks between the rows 21 and 39 after the forward substitution. Also the block C_{21} is non-zero.

The calculation procedure for this situation is described in the step 15.

Row 41: Block $A_{41,20}$:

Since this situation is similar to that of the row 40, the calculation procedure followed is the same. However, it gets simplified to step 16 since the block C_{20} is zero, and therefore the blocks between rows 20 and 40 do not enter in the computations.

Thus, the numerous possible arrangements of off-tri-diagonal blocks which require different computational efforts, and the presence of some null blocks which may change the computations substantially, make it extremely difficult to develop a 'universal' algorithm which can handle all possible arrangements.

CHAPTER 3

SPARSITY EXPLOITATION

The conventional and modified Thomas algorithms exploit the sparsity of the Jacobian matrix which is highly sparse, and has a completely or almost-tridiagonal-band structure.

Moreover, the submatrices within the Jacobian are also sparse (Particularly A and C), and have a definite structure (refer 2.19, 2.20 and 2.21). The presence of zero and identity blocks within the submatrices and their definite structures, allow a further exploitation of the sparsity, which consequently results in a substantial reduction of both computational and storage requirements.

In this chapter, the exploitation of sparsity within the submatrices along with saving in storage and operation-count, is presented.

As is customary, only the operations involving multiplications and divisions are counted, neglecting those involving addition and subtraction since the time consumed in these is comparatively negligible.

To take the advantage of the sparsity and structure of the submatrices, computation and storage involving zero and unity blocks is avoided, which results in a substantial reduction in the CPU-time and memory-requirement on a computer.

Listed below are the various operations involving multiplication and inversion of submatrices, while exploiting their sparsity and structures, and comparison of the operation counts.

It may be pointed out that the P-blocks have two different structures viz. P_V and P_L, which are as under:

$$P^V = \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O} & \bar{X}_c & X \end{bmatrix} \quad ; \quad P^L = \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix}$$

Also, $B^I = B^{-1}$ or $(B-A\ P^V)^{-1}$, where both have the same structure

$$= \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X} \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X} \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix}$$

Operation 1: $P^V \leftarrow B^I C$

$$\begin{bmatrix} \bar{O} & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{O}}_c & r\ \bar{I}_c & \bar{O}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{O}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix}$$

$$P^V_{i,k+c} \leftarrow r\ B^I_{i,k} + \sum_{m=c+1}^{2c+1} B_{i,m}\ C_{m,k+c} \qquad \begin{array}{l} 1 \leq i \leq 2c+1 \\ 1 \leq k \leq c \end{array}$$

$$P^{V}_{i,2c+1} \leftarrow B^{I}_{i,2c+1} \quad C_{2c+1,2c+1} \qquad 1 \leq i \leq 2c+1$$

$$\begin{aligned} \text{S.O.C.}^{*} &= 8C^3 + 12C^2 + 6C + 1 \\ \text{I.O.C.}^{+} &= 2C^3 + 5C^2 + 4C + 1 \\ \text{Saving} &= 6C^3 + 7C^2 + 2C \end{aligned}$$

Operation 2: $P^{V} \leftarrow B^{I}A$

$$\begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{0}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{0}}_c & \bar{X}_c \\ \bar{X}_c & \bar{0}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} r\bar{\bar{I}}_c & \bar{\bar{0}}_c & \bar{0}_c \\ \bar{\bar{0}}_c & \bar{\bar{0}}_c & \bar{0}_c \\ \bar{X}_c & \bar{0}_c & X \end{bmatrix}$$

$$P^{V}_{i,k+c} \leftarrow r\, B^{I}_{i,k} + B^{I}_{i,2c+1} \; A_{2c+1,k} \qquad \begin{aligned} &1 \leq i \leq 2c+1 \\ &1 \leq k \leq c \end{aligned}$$

$$P^{V}_{i,2c+1} \leftarrow B^{I}_{i,2c+1} \; A_{2c+1,2c+1} \qquad 1 \leq i \leq 2c+1$$

$$\begin{aligned} \text{S.O.C.} &= 8C^3 + 12C^2 + 6C + 1 \\ \text{I.O.C.} &= 4C^2 + 4C + 1 \\ \text{Saving} &= 8C^3 + 8C^2 + 2C \end{aligned}$$

---

* Standard Operation Count

+ Improved Operation Count.

Operation 3: $R \leftarrow A\,P^V$

$$\begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} r\bar{\bar{I}}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{\bar{O}}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix}$$

$$R_{i,k+c} \leftarrow r\,P^V_{i,k+c} \qquad 1 \le i \le c, \quad 1 \le k \le c+1$$

$$R_{2c+1,k+c} \leftarrow A_{2c+1,2c+1}\,P^V_{2c+1,k+c} + \sum_{m=1}^{c} A_{2c+1,m}P^V_{m,k+c}$$

$$1 \le k \le c+1$$

S.O.C. $= 8C^3 + 12C^2 + 6C + 1$

I.O.C. $= 2C^2 + 2C + 1$

Saving $= 8C^3 + 10C^2 + 4C$

Operation 4: $R \leftarrow A\,P^L$

$$\begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{I}}_c & I\bar{\bar{O}}_c & \bar{O}_c \\ \bar{\bar{O}}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix}$$

$$R_{i,k} \leftarrow r\;P^L_{i,k} \qquad 1 \le i \le c, \quad 1 \le k \le c+1$$

$$R_{2c+1,k} \leftarrow A_{2c+1,2c+1} P^L_{2c+1,k} + \sum_{m=1}^{c} A_{2c+1,m} P^L_{m,k} \quad 1 \leq k \leq c$$

$$R_{2c+1,2c+1} \leftarrow A_{2c+1,2c+1} P^L_{2c+1,2c+1} + \sum_{m=1}^{c} A_{2c+1,m} P^L_{m,2c+1}$$

$$\begin{aligned} \text{S.O.C.} &= 8C^3 + 12C^2 + 6C + 1 \\ \text{I.O.C.} &= 2C^2 + 2C + 1 \\ \text{Saving} &= 8C^3 + 10C^2 + 4C \end{aligned}$$

Operation 5: $R \leftarrow C\, P^V$

$$\begin{bmatrix} \bar{\bar{0}} & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{0}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{0}_c & \bar{X}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{0}}_c & r\bar{\bar{I}}_c & \bar{0}_c \\ \bar{\bar{0}}_c & \bar{\bar{X}}_c & \bar{0}_c \\ \bar{\bar{0}}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{0}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{0}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{0}}_c & \bar{\bar{X}}_c & X \end{bmatrix}$$

$$R_{i,k+c} \leftarrow r\, P^V_{i+c,k+c} \qquad 1 \leq i \leq c,\ 1 \leq k \leq c+1$$

$$R_{i+c,k+c} \leftarrow \sum_{m=c+1}^{2c} C_{i+c,m} P^V_{m,k+c} \quad 1 \leq i \leq c, 1 \leq k \leq c+1$$

$$R_{2c+1,k+c} \leftarrow \sum_{m=c+1}^{2c+1} C_{2c+1,m} P^V_{m,k+c} \quad 1 \leq k \leq c+1$$

$$\begin{aligned} \text{S.O.C.} &= 8C^3 + 12C^2 + 6C + 1 \\ \text{I.O.C.} &= C^3 + 3C^2 + 3C + 1 \\ \text{Saving} &= 7C^3 + 9C^2 + 3C \end{aligned}$$

Operation 6: $R \leftarrow C\,P^{L}$

$$\begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{0}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{0}}_c & \bar{X}_c \\ \bar{X}_c & \bar{0}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{0}}_c & r\bar{\bar{I}}_c & \bar{\bar{0}}_c \\ \bar{\bar{0}}_c & \bar{\bar{X}}_c & \bar{0}_c \\ \bar{0}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{0}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{0}}_c & \bar{X}_c \\ \bar{X}_c & \bar{0}_c & X \end{bmatrix}$$

$$R_{i,k} \leftarrow r\,P^{L}_{i+c,k} \qquad 1 \le i \le c,\ 1 \le k \le c$$

$$R_{i,2c+1} \leftarrow r\,P^{L}_{i+c,2c+1} \qquad 1 \le i \le c$$

$$R_{i+c,k} \leftarrow \sum_{m=c+1}^{2c} C_{i+c,m}\,P^{L}_{m,k} \qquad 1 \le i \le c,\ 1 \le k \le c$$

$$R_{i+c,2c+1} \leftarrow \sum_{m=c+1}^{2c} C_{i+c,m}\,P^{L}_{m,2c+1} \qquad 1 \le i \le c$$

$$R_{2c+1,k} \leftarrow \sum_{m=c+1}^{2c+1} C_{2c+1,m}\,P^{L}_{m,k} \qquad 1 \le k \le c$$

$$R_{2c+1,2c+1} \leftarrow \sum_{m=c+1}^{2c+1} C_{2c+1,m}\,P^{L}_{m,2c+1}$$

S.O.C. $= 8C^3 + 12C^2 + 6C + 1$

I.O.C. $= C^3 + 3C^2 + 3C + 1$

Saving $= 7C^3 + 9C^2 + 3C$

Operation 7: $P^V \leftarrow B^I R$, where $R = -AP^V$ (refer OP 3)

$$\begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{O}}_c & \bar{\bar{O}}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix}$$

$$P^V_{i,k+c} \leftarrow B^I_{i,2c+1}\, R_{2c+1,k+c} + \sum_{m=1}^{c} B^I_{i,m}\, R_{m,k+c}$$

$1 \leq i \leq 2c+1, 1 \leq k \leq c+1$

$S.O.C. = 8C^3 + 12C^2 + 6C + 1$

$I.O.C. = 2C^3 + 5C^2 + 4C + 1$

$\text{Saving} = 6C^3 + 7C^2 + 2C$

Operation 8: $P^L \leftarrow B^I R$, where $R = -A P^L$ (refer OP 4)

$$\begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{O}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix}$$

$$P^L_{i,k} \leftarrow B^I_{i,2c+1}\, R_{2c+1,k} + \sum_{m=1}^{c} B_{i,m}\, R_{m,k} \quad 1 \leq i \leq 2c+1, 1 \leq k \leq c$$

$$P^L_{i,2c+1} \leftarrow B^I_{i,2d+1}\, R_{2c+1,2c+1} + \sum_{m=1}^{c} B_{i,m}\, R_{m,2c+1} \quad 1 \leq i \leq 2c+1$$

$S.O.C. = 8C^3 + 12C^2 + 6C+1$

$I.O.C. = 2C^3 + 5C^2 + 4C+1$

$\text{Saving} = 6C^3 + 7C^2 + 2C$

Operation 9: $P^V \leftarrow B^I R$, where $R = C\ P^V$ (refer OP 5)

$$\begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix}$$

$$P^V_{i,k+c} \leftarrow \sum_{m=1}^{2c+1} B^I_{i,m}\ R_{m,k+c} \qquad 1 \leq i \leq 2c+1,\ 1 \leq k \leq c+1$$

$$\begin{aligned} S.O.C. &= 8C^3 + 12C^2 + 6C + 1 \\ I.O.C. &= 4C^3 + 8C^2 + 5C + 1 \\ \text{Saving} &= 4C^3 + 4C^2 + C \end{aligned}$$

Operation 10: $P^L \leftarrow B^I R$ where $R = C\ P^L$ (refer OP 6)

$$\begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix}$$

$$P^L_{i,k} \leftarrow \sum_{m=1}^{2c+1} B^I_{i,m}\ R_{m,k} \qquad 1 \leq i \leq 2c+1,\quad 1 \leq k \leq c$$

$$P^L_{i,2c+1} \leftarrow \sum_{m=1}^{2c+1} B^I_{i,m}\ R_{m,2c+1} \qquad 1 \leq i \leq 2c+1$$

$$\begin{aligned} S.O.C. &= 8C^3 + 12C^2 + 6C + 1 \\ I.O.C. &= 4C^3 + 8C^2 + 5C + 1 \\ \text{Saving} &= 4C^3 + 4C^2 + C \end{aligned}$$

Operation 11 : $R \leftarrow P^V P^L$

$$\begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix}$$

$$R_{i,k} \leftarrow \sum_{m=c+1}^{2c+1} P^V_{i,m} \; P^L_{m,k} \qquad 1 \leq i \leq 2c+1, \; 1 \leq k \leq c$$

$$R_{i,2c+1} \leftarrow \sum_{m=c+1}^{2c+1} P^V_{i,m} \; P^L_{m,2c+1} \qquad 1 \leq i \leq 2c+1$$

S.O.C. $= 8C^3 + 12C^2 + 6C + 1$

I.O.C. $= 2C^3 + 5C^2 + 4C + 1$

Saving $= 6C^3 + 7C^2 + 2C$

Operation 12: $R \leftarrow P^L P^L$

$$\begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix}$$

$$R_{i,k} \leftarrow P^L_{i,2c+1} \; P^L_{2c+1,k} + \sum_{m=1}^{c} P^L_{i,m} \; P^L_{m,k} \qquad 1 \leq i \leq 2c+1, 1 \leq k \leq c$$

$$R_{i,2c+1} \leftarrow P^L_{i,2c+1} \; P^L_{2c+1,2c+1} + \sum_{m=1}^{c} P^L_{i,m} \; P^L_{m,2c+1} \qquad 1 \leq i \leq 2c+1$$

S.O.C. $= 8C^3 + 12C^2 + 6C + 1$

I.O.C. $= 2C^3 + 5C^2 + 4C + 1$

Saving $= 6C^3 + 7C^2 + 2C$

Operation 13: $R \leftarrow P^L P^V$

$$\begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix}$$

$$R_{i,k+c} \leftarrow P^L_{i,2c+1} \; P^V_{2c+1,k+c} + \sum_{m=1}^{c} P^L_{i,m} \; P^V_{m,k+c}$$

$1 \leq i \leq 2c+1, \; 1 \leq k \leq c+1$

S.O.C. = $8C^3 + 12C^2 + 6C + 1$

I.O.C. = $2C3 + 5C^2 + 4C + 1$

Saving = $6C^3 + 7C^2 + 2C$

Operation 14: $R \leftarrow P^V \; P^V$

$$\begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix}$$

$$R_{i,k+c} \leftarrow \sum_{m=c+1}^{2c+1} P^V_{i,m} \; P^V_{m,k+c} \qquad 1 \leq i \leq 2c+1, \; 1 \leq k \leq c+1$$

S.O.C. = $8C^3 + 12C^2 + 6C+1$

I.O.C. = $2C^3 + 5C^2 + 4C + 1$

Saving = $6C^3 + 7C^2 + 2C$

Operation 15: $S \leftarrow AQ$

$$\begin{bmatrix} \bar{X}_c \\ \bar{O}_c \\ X_c \end{bmatrix} \leftarrow \begin{bmatrix} r\bar{\bar{I}}_c & \bar{O}_c & \bar{O}_c \\ \bar{\bar{O}}_c & \bar{\bar{O}}_c & \bar{O}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix}$$

$S_i \leftarrow r\,Q_i, \quad 1 \leq i \leq c$

$S_{2c+1} \leftarrow A_{2c+1,2c+1}\; Q_{2c+1} + \sum_{m=1}^{c} A_{2c+1,m}\; Q_m$

S.O.C. $= 4C^2 + 4C + 1$

I.O.C. $= 2C + 1$

Saving $= 4C^2 + 2C$

Operation 16: $S \leftarrow C\,Q$

$$\begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{O}}_c & r\bar{\bar{I}}_c & \bar{O}_c \\ \bar{\bar{O}}_c & \bar{X}_c & \bar{O}_c \\ \bar{O}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix}$$

$S_i \leftarrow r\;Q_{i+c} \qquad 1 \leq i \leq c$

$S_{i+c} \leftarrow \sum_{m=c+1}^{2c} C_{i+c,m}\; Q_m \quad 1 \leq i \leq c$

$S_{2c+1} \leftarrow \sum_{m=c+1}^{2c+1} C_{2c+1,m}\; Q_m$

S.O.C. $= 4C^2 + 4C + 1$

I.O.C. $= C^2 + 2C + 1$

Saving $= 3C^2 + 2C$

Operation 17: $S \leftarrow P^V Q$

$$\begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{\bar{O}}_c & \bar{\bar{X}}_c \\ \bar{O}_c & \bar{X}_c \end{bmatrix} \begin{matrix} \bar{X}_c \\ \bar{X}_c \\ X \end{matrix} \quad \begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix}$$

$$S_i \quad \sum_{m=c+1}^{2c+1} P^V_{i,m} \; Q_m \qquad 1 \leq i \leq 2c+1$$

S.O.C. $= 4C^2 + 4C + 1$

I.O.C. $= 2C^2 + 3C + 1$

Saving $= 2C^2 + C$

Operation 18: $S \leftarrow P^L Q$

$$\begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{O}}_c & \bar{X}_c \\ \bar{X}_c & \bar{O}_c & X \end{bmatrix} \begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix}$$

$$S_i \leftarrow P^L_{i,2c+1} \; Q_{2c+1} + \sum_{m=1}^{c} P_{i,m} \; Q_m \qquad 1 \leq i \leq 2c+1$$

S.O.C. $= 4C^2 + 4c + 1$

I.O.C. $= 2C^2 + 3C + 1$

Saving $= 2C^2 + C$

Operation 19: $S \leftarrow B^I F$ (Standard Operation)

$$\begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{bmatrix} \begin{bmatrix} \bar{X}_c \\ \bar{X}_c \\ X \end{bmatrix}$$

$$S_i \leftarrow \sum_{m=1}^{2c+1} B^I_{i,m} \; F_m \qquad 1 \leq i \leq 2c+1$$

S.O.C. = $4C^3 + 4C^2 + 1$

I.O.C. = $4C^3 + 4C^2 + 1$

Saving = 0

Operation 20: Inversion of B or (B − AP)

The B and (B− AP) matrices have the same structure, which is as under:

$$\left[\begin{array}{c|cc} -(1+s)\bar{I}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \hline \bar{\bar{X}}_c & \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & \bar{X}_c & X \end{array}\right] = \left[\begin{array}{c|c} \bar{\bar{b}}_{11} & \bar{\bar{b}}_{12} \\ \hline \bar{\bar{b}}_{21} & \bar{\bar{b}}_{22} \end{array}\right]$$

Inversion by partitioning does not result in saving of the computational effort. It may be noted here that this matrix contains a cxc identity matrix, I_c, and hence some saving in the operations may be achieved as demonstrated below.

We partition the matrix as shown above (one partion only)

Hence, B^{-1} or $(B - AP)^{-1} = B^{I} = \left[\begin{array}{c|c} \bar{\bar{D}}_{11} & \bar{\bar{D}}_{12} \\ \hline \bar{\bar{D}}_{21} & \bar{\bar{D}}_{22} \end{array}\right]$

where,

order of $\bar{\bar{D}}_{11} = c \times c$

order of $\bar{\bar{D}}_{12} = c \times (c+1)$

order of $\bar{\bar{D}}_{21} = (c+1) \times c$

order of $\bar{\bar{D}}_{22} = (c+1) \times (c+1)$

and,

$$\bar{\bar{D}}_{22} = (\bar{\bar{b}}_{22} - \bar{\bar{b}}_{21} (\bar{\bar{b}}_{11})^{-1} \bar{\bar{b}}_{12})^{-1} \qquad (1)$$

$$\bar{\bar{D}}_{11} = (\bar{\bar{b}}_{11})^{-1} + (\bar{\bar{b}}_{11})^{-1} \bar{\bar{b}}_{12} \bar{\bar{D}}_{22} \bar{\bar{b}}_{21} (\bar{\bar{b}}_{21})^{-1} \qquad (2)$$

$$\bar{\bar{D}}_{12} = -(\bar{\bar{b}}_{11})^{-1} \bar{\bar{b}}_{12} \bar{\bar{D}}_{22} \qquad (3)$$

$$\bar{\bar{D}}_{21} = -\bar{\bar{D}}_{22} \bar{\bar{b}}_{21} (\bar{\bar{b}}_{11})^{-1} \qquad (4)$$

Since $(b_{11})^{-1} = -\frac{1}{(1+s)} \bar{\bar{I}}_c$, (only one division is needed to compute it), we need to perform only one inversion of the $(c+1) \times (c+1)$ in equation (1), to compute the B^{I}.

S.O.C. $= 8C^3 + 12C^2 + 6C + 1$

I.O.C. $= 5C^3 + 12C^2 + 8C + 2$

Saving $= 3C^3 - (2c+1)$

$\approx 3C^3$

In the case of a single column problem with no pump-around or bypass streams, only the operations 1,3,15,17,19 and 20 are performed in the conventional Thomas algorithm.

Moreover, the fractions r_{Vj} and r_{Lj} are unity for all the trays, since there are no interlinked-streams present. Therefore, in the operations 1,3 and 15 there is no need to multiply these fractions and a further reduction in the operation count is obtained. The improved operation count for these operations is presented below:

	Improved operation count
Operation 1 :	$2C^3 + 3C^2 + 3C + 1$
Operation 3 :	$C^2 + 2C + 1$
Operation 15 :	$C + 1$

Saving in Storage Requirement:

In the conventional or modified Thomas algorithm, computations are performed stage by stage. Since the computations for any row j, require only tridiagonal and off-diagonal P-blocks of the previous j-1 rows, the A, B and C submatrices need not be stored, nor the identity blocks on the diagonal after the forward substitution. Thus, we need to store only the P-blocks. Since in both types of P-blocks, there are (2c+1) x c zero elements which need not be stored.

$$P^V = \left[\begin{array}{c} 0_c \\ \bar{\bar{0}}_c \\ \bar{0}_c \end{array}\left[\begin{array}{cc} \bar{\bar{X}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{X}_c \\ \bar{X}_c & X \end{array}\right]\right. \qquad P^L = \left[\begin{array}{c:c:c} \bar{\bar{X}}_c & \bar{\bar{0}}_c & \bar{X}_c \\ \bar{\bar{X}}_c & \bar{\bar{0}}_c & \bar{X}_c \\ \bar{X}_c & \bar{0}_c & X \end{array}\right]$$

Thus, out of $(2c+1) \times (2c+1)$ elements of any P-block only $(2c+1) \times (c+1)$ are stored, and thereby almost a 50% further saving in storage results (Refer Chapter 6).

CHAPTER 4

SPECIFICATIONS

A multicomponent, multistage separation process problem must have the following specifications:

(i) Number of trays

(ii) Number of components

(iii) Column pressure

(iv) Complete specification of the feed (the feed rate, composition, thermal conditions, and the location of the feed tray).

In case of distillation, the location and type of every condenser and reboiler must also be specified.

In addition to these, some more specifications are needed to completely define the problem. These specifications may be made for the condensers, reboilers or for any other intermediate stages. It is important that the total number of these specifications must be equal to the total number of condensers and reboilers present in the system, otherwise the problem would become under or overspecified.

Specifications for Condensers and Reboilers:

In the Naphtali-Sandholm model, the specifications for condensers and reboilers are their heat duties, which are incorporated in the enthalpy balance discrepancy function, E_j.

For specifications other than the heat duties, the enthalpy balance in equations (2.27), (2.30), (2.33) or (2.36) shall no longer be applicable, and should therefore be replaced by the suitable discrepancy functions which takes into account the given specifications.

Some of these specifications and their corresponding discrepancy functions are presented in this chapter.

Specifications for the Intermediate Trays

When specifications are made at the intermediate stages other than condensers and reboilers, the heat-duties are no longer variable, and get fixed accordingly. Consequently, the enthalpy balance discrepancy functions, E_j, for the condensers and reboilers are no longer applicable.

Hofeling and Seader [2], have proposed that the inapplicable discrepancy function E_j, for a condenser or reboiler tray, may be replaced by a suitable discrepancy function, which incorporates the specification at the given intermediate stage. This replacement however generates an off-tridiagonal block in the Jacobian since the function E_j for a condenser or reboiler contains the variable(s) of an intermediate stage. The modified Thomas algorithm may be employed to handle the off-tridiagonal blocks.

Generation of the off-tridiagonal blocks results in additional computational and storage requirements, and should therefore be avoided, if possible. It is therefore proposed here, that the discrepancy function E_j for condenser or reboiler may be modified and made applicable (as in Spc. 1 and Spc. 2 below), and the E_j for the intermediate stage should be replaced by a suitable discrepancy function to incorporate the given specification at the same tray.

Listed below are some specifications and corresponding discrepancy functions which may be used to replace the inapplicable ones:

Spc. 1: If a specification is made at an intermediate stage instead of condenser, the condenser heat duty Q_c in equations 2.27 and 2.30 cannot be specified and should therefore be computed. Calculations of Q_c, and the modified E_j are as under:

Partial Condenser: In a partial condenser, (which is represented by tray j), out of the r_{vj+1}, V_{j+1} moles of the vapor stream, $(1 + s_j)L_j$ moles are condensed (Figure 3). Therefore, Q_c may be computed by enthalpy considerations:

$$Q_c = -(1 + s_j) \sum_{i=1}^{c} (H_{j+1,i} - h_{j,i}) l_{j,i}$$

and

$$E_j = \sum_{i=1}^{c} H_{j+1,i} (r_{Vj+1} v_{j+1,i} - (1+s_j) l_{j,i}) -$$

$$- (1 + S_j) \sum_{i=1}^{c} H_{j,i} v_{j,i}$$

Total condenser: In a total condenser, tray j, the whole $r_{vj+1} V_{j+1}$ stream is condensed.

$$Q_c = -r_{vj+1} \sum_{i=1}^{c} (H_{j+1,i} - h_{j,i}) v_{j+1,i}$$

$$E_j = \sum_{i=1}^{c} h_{j,i} (r_{vj+1} v_{j+1,i} - (1+s_j) l_{j,i} - (1+S_j) v_{j,i})$$

Spc. 2: If a specification is made at an intermediate stage instead of a reboiler, the reboiler heat duty Q_R in equation (2.33) and (2.36), cannot be specified and should therefore be computed.

Partial Reboiler: In a partial reboiler (which is represented by tray j), out of the $r_{vj+1} L_{j-1}$ moles of the liquid stream, $(1 + S_j)V_j$ moles are vaporized (Figure 3). Q_R may be computed by enthalpy considerations:

$$Q_R = (1 + S_j) \sum_{i=1}^{c} (H_{j,i} - h_{j-1,i}) v_{j,i}$$

$$E_j = \sum_{i=1}^{c} h_{j-1,i} (r_{vj-1}\, l_{j-1,i} - (1 + S_j) v_{j,i}) - $$
$$ - (1 + s_j) \sum_{i=1}^{c} h_{j,i}\, l_{j,i}$$

Total Reboiler: In this case, out of $r_{vj-1}\, L_{j-1}$ moles, $(1 + s_j)l_j$ moles are withdrawn and remaining $(1 + S_j)V_j$ are fed to the reboiler and are completely vaporized (Figure 3).

$$Q_R = (1 + S_j) \sum_{i=1}^{c} (H_{j,i} - h_{j-1,i}) v_{j,i}$$

$$E_j = \sum_{i=1}^{c} h_{j-1,i} (r_{vj-1}\, l_{j-1,i} - (1+s_j) l_{j,i} - (1+S_j) v_{j,i})$$

Spc. 3: Specification of the condenser-heat-duty, Q_c. As discussed earlier, (Please refer equations (2.27) and (2.30) for E_j).

Spc. 4: Specification of the reboiler-heat-duty, Q_R. As discussed earlier (Please refer equations (2.33) and (2.36) for E_j)

Spc. 5: Specification of the Reflux Ratio, $R = L/D$ for a condenser. (or $R = (L/V)$ may be specified for an intermediate stage)

$$E_j = L_j - R\, V_j$$

Spc. 6: Specification of the Reboil Ratio, $r = V/B$ for a reboiler. (or $r = (V/L)$ may be specified for an intermediate stage) $E_j = V_j - r\, L_j$.

Spc. 7: Specification of Temperature at any stage j

$E_j = T_j - T_{spec.}$

Spc. 8: Specification of total vapor flow rate at any stage j.

(V_{spec} = D = Distillate rate, for a condenser)

$E_j = V_j - V_{spec}$.

Spc. 9: Specification of total liquid flow rate at any stage j.

(L_{spec} = B = Bottoms rate, for a reboiler)

$E_j = L_j - L_{spec}$.

Spc.10: Specification of molar vapor flow rate of component i at a stage j. ($(v_{j,i})_{spec.} = d_i$ = vapor key component flow rate, for a condenser tray).

$E_j = v_{j,i} - (v_{j,i})_{spec}$.

Spc.11: Specification of molar liquid flow rate of component i at a stage j. ($(l_{j,i})_{spec.} = b_i$ = liquid key component flow rate, for a reboiler tray).

$E_j = l_{j,i} - (l_{j,i})_{spec}$.

Spc.12: Specification of vapor mole fraction of component i at stage j. ($(y_{j,i})_{spec.} = x_{d_i}$ = vapor key component composition for a condenser tray).

$E_j = v_{j,i} - V_j(y_{j,i})_{spec.}$

Spc.13: Specification of liquid mole fraction of component i at stage j. ($(x_{j,i})_{spec} = x_{b_i}$ = liquid key component composition for a reboiler tray)

$E_j = l_{j,i} - L_j(x_{j,i})_{spec}$.

Consider a single distillation column with 15 trays (including the condenser as tray 1 and reboiler as tray 15), and a 6-component mixture. Instead of condenser and reboiler heat-duties, the temperature of tray 9 and the flow rate of vapor leaving the tray 10 are specified.

As proposed by Hofeling and Seader in [2], the condenser enthalpy balance discrepancy function E_1, is replaced by $E_1 = T_9 - (T_9)_{specified}$, and E_{15} of reboiler by $E_{15} = V_{10} - (V_{10})_{specified}$. The resulting Jacobian structure is shown in Figure 8, which may be solved using the modified Thomas algorithm.

Alternatively, we can modify the discrepancy functions E_1 and E_{15} as discussed in Spc. 1 and Spc. 2, and replace those of rows 9 and 10 by $E_9 = T_9 - (T_9)_{specified}$ and $E_{10} = V_{10} - (V_{10})_{specified}$, respectively. The resulting Jacobian matrix, in this case, has a completely tridiagonal structure with no off-tridiagonal elements. The conventional Thomas algorithm is employed here, and a substantial reduction in the computational efforts (3.877 times), and in memory requirements (2.786 times) is thereby realized.

CHAPTER 5

IMPLEMENTATION OF THE ALGORITHM

A computer program for solving the multicomponent separation process problems has been written in FORTRAN 10 and implemented on DEC 1090 system based on the algorithm described earlier. The program is kept in a file SEP.FOR.

The file SEP.FOR consists of a main program INTLNK and various subroutines and the tasks performed by each of them are listed below:

Main Program:

INTLNK: The program may be considered to have two parts with different tasks. The first part reads the data, and computes the discrepancy function - vector. This part has been written for a general system of interlinked-columns and can handle any arbitrary arrangement of columns.

The second part includes the algorithm for solving a specific problem. As has been pointed out in Chapter 2, it is difficult to develop a generalized Hofeling-Seader algorithm which can handle all possible arrangements of the off-tridiagonal blocks in a Jacobian. Therefore this part must be rewritten for a given arrangement of the off-tridiagonal blocks.

Subroutines

ENL	:	Computes the liquid-molar-enthalpy for all components at a given stage (using a polynomial).
ENV	:	Computes the vapor-molar-enthalpy for all components at a given stage (using a polynomial).
FINDK	:	Computes the equilibrium K_{ij} values for all components for a given stage (using Antoine's constants (ideal-case)).
DKBYDX	:	Computes the derivatives $\frac{\partial k}{\partial x}$
DKBYDY	:	Computes the derivatives $\frac{\partial k}{\partial y}$
DKBYDT	:	Computes the derivatives $\frac{\partial k}{\partial T}$
DHVDT	:	Computes the derivatives $\frac{\partial H}{\partial T}$
DHLDT	:	Computes the derivatives $\frac{\partial h}{\partial T}$
CONBC	:	Computes the elements of the B & C submatrices for a partial or a total condenser.
REBAB	:	Computes the elements of the A & B submatrices for a partial or a total reboiler.
TRIA	:	Computes the elements of the A submatrix for a given tray.
TRIB	:	Computes the elements of the B submatrix for a given tray.
TRIC	:	Computes the elements of the C submatrix for a given tray.

OFFA	:	Computes the elements of the off-tridiagonal block having the same structure as that of an A-submatrix. This block is generated because of the liquid interlinked-stream.
OFFC	:	Computes the elements of the off-tridiagonal blocks having the same structure as that of C-matrix. This block is generated because of the vapor-interlinked stream.
PROB	:	Replaces the last row of the A, B and C sub-matrices to incorporate the given specification for a stage j, where j may represent a condenser, reboiler or an intermediate stage.

The following subroutines perform the various operations, which have been presented in the Chapter),

BCMUL	:	Operation 1
BAMUL	:	Operation 2
APVMUL	:	Operation 3
APLMUL	:	Operation 4
CPVMUL	:	Operation 5
CPLMUL	:	Operation 6
MBAPV	:	Operation 7
MBAPL	:	Operation 8
BCPV	:	Operation 9

BCPL	:	Operation 10
PVPL	:	Operation 11
PLPL	:	Operation 12
PLPV	:	Operation 13
PVPV	:	Operation 14
AQMUL	:	Operation 15
CQMUL	:	Operation 16
QMPVX	:	Operation 17
QMPLX	:	Operation 18
BFMUL	:	Operation 19
INVPRT	:	Operation 20 (This subroutine computes the inverse of a B or (B-AP) matrix of order (2c+1) x (2c+1))
MATINV	:	Computes the inverse of (c+1)x(c+1) matrix using Gauss Jordan method with the maximum pivot strategy (called by INVPRT).

After the input-data has been read, a check is conducted to find whether the problem is under or over specified, and if found so the execution is immediately terminated with an error message. Several checks are made at the various stages of execution, and if any inconsistency is detected, a suitable error message is printed out and the execution is stopped.

For a single column problem, the variable NOFF, which represents the number of off-tridiagonal blocks in the Jacobian, should be set to zero, and the program uses the conventional Thomas algorithm to solve the problem.

The thirteen different specifications for the condensers, reboilers and intermediate trays, which are presented in the Chapter 4, have been implemented in the program. For a given specification, a suitable code (1 through 13) is assigned to the variable LABEL and the specifications are handled following the approach discussed earlier. The subroutine PROB modifies the corresponding derivatives in the last row of the submatrices A, B, and C to incorporate the given specification.

In a system of interlinked columns a few vapor or liquid streams may be missing. This situation may arise in the cases where splitting and rearrangement of columns is done to obtain the "best-ordering" [4] . The locations of the trays for which a leaving-vapor stream is missing are stored in the JNOSV array, and those for the missing liquid streams in the JNOSL array. The missing streams are omitted in the calculations.

The fractions r_V and r_L (refer Chapter 2), are unity for the trays from which no interlinked stream is leaving, and for all single column problems. Therefore these need not be included in the various multiplication operations such as operation 1,2 and 3 etc. The subroutines take care of this fact and a further saving in the operation count may be realized.

CHAPTER 6

RESULTS AND DISCUSSIONS

The exploitation of sparsity of the submatrices, while performing the matrix multiplications and inversions in the algorithms presented in Chapter 2, has resulted in a significant reduction in computational and storage requirements. The reduction in both, the operation count and the storage, for a variety of problems is presented in this chapter.

Problem 1: Consider a single absorption column having 8 stages and using a mixture of 14-components (For the complete specifications please refer [9]).

The problem was solved by applying the conventional Thomas algorithm, and a comparison of the operation counts is made in the Table 1.

A substantial reduction in the computation by a ratio of 3.02 was obtained by the sparsity exploitation, as only 183774 operations were performed instead of 555060.

Since only 3045 (= 7x29x14) elements need to be stored instead of 5887 (= 7x29x29), a saving of about 48% in the storage was realized.

This problem when implemented on DEC-10 took 4 iterations to converge to a tolerance limit of 9×10^{-4}

(sum of squares of all the discrepancy functions). The actual CPU time spent in the matrix multiplications and inversions were 32.59 and 11.61 seconds for the standard and improved operations respectively. Thus the ratio of the CPU time is about 2.81 as compared to the theoretically obtained ratio of 3.02.

The total CPU time spent in the execution of the programs for the standard and improved operations are 38.43 and 17.43 seconds respectively, and the balance was spent in evaluating the discrepancy functions, elements of the submatrices, and in executing the various input and output statements.

With the original discrepancy functions, proposed by Naphtali and Sandholm (refer equation 2.10), this program took 7 iterations to converge, whereas only 4 iterations were needed with the modified equation 2.2. Thus, this minor change has improved the convergence characteristics.

It was realized that the large roundoff errors in the inversion of the submatrices may impair the convergence characteristics. Three different subroutines used to study the convergence characteristics, and the results of which are as under:

Matrix inversion subroutine	Number of iterations required
1. Gauss Jordan with the maximum pivot-strategy	4
2. Gauss elimination	5
3. FO1AAF/NAG (available on DEC-10)	12

A check on the roundoff errors revealed that the FO1AFF subroutine resulted in substantially large roundoff errors, in the matrix inversions.

It was also noted that the inversion by partitioning method resulted in less roundoff errors as compared to the standard matrix inversion, particularly in the matrices where the difference in the magnitude of the elements was vast.

Problem 2: Consider a single absorption column having 20 stages with a mixture of 4 components. (For the complete specifications please refer [1]).

This problem was solved using the conventional Thomas algorithm and a comparison of the operation counts is made in the Table 2.

A reduction in the computations by a ratio of 2.676 is obtained by sparsity exploitation, as only 17556 operations were performed instead of 46980.

This problem when implemented on DEC-10, took 4 iterations to converge to a tolerance limit of $5x10^{-8}$

TABLE 2

Comparison of Operation Counts for Problem 2

Number of components = 4

Number of stages = 20

Operations	Number of times performed	Standard operation count	Improved operation count
Op 1	19	x729 = 13851	x189 = 3591
Op 3	19	x729 = 13851	x 25 = 475
Op 15	19	x 81 = 1539	x 5 = 95
Op 17	19	x 81 = 1539	x 45 = 855
Op 19	20	x 81 = 1620	x 81 = 1620
Op 20	20	x 729 = 14580	x546 = 10920
Total		46980	17556

$$\frac{\text{S.O.C.}}{\text{I.O.C.}} = 2.676$$

(sum of squares of all discrepancy functions). The total CPU time spent in the matrix multiplications and inversions were 2.82 and 1.22 sec. for the standard and improved operations respectively. Thus the ratio of the CPU time is about 2.31 as against the theoretically obtained ratio of 2.676.

The total CPU time for the standard and improved programs was 7.02 and 4.34 seconds respectively, and the balance was spent in the evaluation of the discrepancy functions, the elements of the submatrices, and in executing the input and output statements.

There are a total of 19 P-blocks in the matrix after the forward substitution, each of order 9x9. Since only 19x9x5 elements of the P-blocks need to be stored instead of 19x9x9 44% saving in the storage is obtained.

<u>Problem 3</u>: Consider a single column with 15 stages and a 6-component mixture. In this case, as discussed in the Chapter 4, instead of specifying the heat duties of the condenser and reboiler, specifications at the 9th and 10th stages were made. The temperature of the stage 9, and the flow rate of the vapor leaving the stage 10 are specified. In the Chapter 4, two alternate approaches to solve this problem have been discussed, and the operation counts for both are compared in the Table 3.

TABLE 3

Comparison of Operation Counts for Problem 3

Number of components = 6
Number of stages =15

Operations	Standard operation count			Improved operation count		
	Number of times perfomed	Count per operation	Operation count	Number of times Op i was perfor-med	Count per Op-i	Operation count for Op i
Op 1	15	2197	32955	14	637	8918
Op 3	22	2197	48334	14	85	1190
Op 7	6	2197	13182			
Op 14	4	2197	8788			
Op 15	15	169	2535	14	13	182
Op 17	25	169	4225	14	91	1274
Op 19	15	169	2535	15	169	2535
Op 20	15	2197	32955	15	1562	23430
TOTAL			145509			37529

$$\frac{S.O.C.}{I.O.C.} = 3.877.$$

For the first approach which was proposed by Hofeling and Seader [2], the standard operation count is computed, for obtaining the solution by the modified Thomas algorithm using the Jacobian structumre shown in the Figure 4.

For the latter approach, the improved operation count with the sparsity exploitation was computed for obtaining the solution by the conventional Thomas algorithm. The Jacobian matrix in this case has a tridiagonal band structure.

The standard and improved operation counts are 145509 and 35729 respectively, which results in a computation reduction ratio of 3.877.

In the Hofeling-Seader approach, there are 21 P-blocks (14 tridiagonal and 7 off-tridiagonal), each of order 13x13. Whereas in the second approach there are no off-diagonal blocks and hence a total of only 14 P-blocks in Jacobian.

Therefore following the latter approach, only 1274 (= 14x13x7) elements need to be stored, whereas the standard storage requirement for the two approaches is 3549 (= 21x13x13) and 2366 (= 14x13x13) respectively.

The proposed approach is therefore more efficient, then the Hofeling-Seader approach since it requires substantially less computations and storage.

$$\begin{bmatrix}
B_1 & C_1 & & & & & & & C_{1,9} & & & & & & \\
A_2 & B_2 & C_2 & & & & & & & & & & & & \\
 & A_3 & B_3 & C_3 & & & & & & & & & & & \\
 & & A_4 & B_4 & C_4 & & & & & & & & & & \\
 & & & A_5 & B_5 & C_5 & & & & & & & & & \\
 & & & & A_6 & B_6 & C_6 & & & & & & & & \\
 & & & & & A_7 & B_7 & C_7 & & & & & & & \\
 & & & & & & A_8 & B_8 & C_8 & & & & & & \\
 & & & & & & & A_9 & B_9 & C_9 & & & & & \\
 & & & & & & & & A_{10} & B_{10} & C_{10} & & & & \\
 & & & & & & & & & A_{11} & B_{11} & C_{11} & & & \\
 & & & & & & & & & & A_{12} & B_{12} & C_{12} & & \\
 & & & & & & & & & & & A_{13} & B_{13} & C_{13} & \\
 & & & & & & & & & & & & A_{14} & B_{14} & C_{14} \\
 & & & & & & & & & A_{15,10} & & & & A_{15} & B_{15}
\end{bmatrix}$$

FIG 4 THE STRUCTURE OF THE JACOBIAN MATRIX (PROBLEM 3)

Problem 4: Consider the system of interlinked columns shown in the Figure 5, in which a 4-component mixture is being separated using two absorbed A_1 and A_2 having 10 plates each, and two distillation columns having 15 and 12 plates respectively. The arrangement of columns and ordering of the trays is also shown in the same figure, and the structure of the Jacobian matrix is presented in the Figure 6. (For the complete specifications please refer Ketchum [3]).

The modified Thomas algorithm was employed to solve the system, and the various steps of which are presented in the Chapter 2.

By exploiting the sparsity a reduction in the computations by a ratio of 3.013 was obtained, as only 62961 operations were performed instead of 189702 standard operations. The comparison and details of the standard and improved operation counts have been presented in the Table 4.

There are 46 upper diagonal, 53 off-tridiagonal, and hence a total of 99 P-blocks in the matrix, after the forward substitution. Each P-block is of the order 9x9. As discussed earlier only a 9x5 matrix for every P-block needs storing, since the remaining elements are zero, and hence a 44% saving in the storage may be realized.

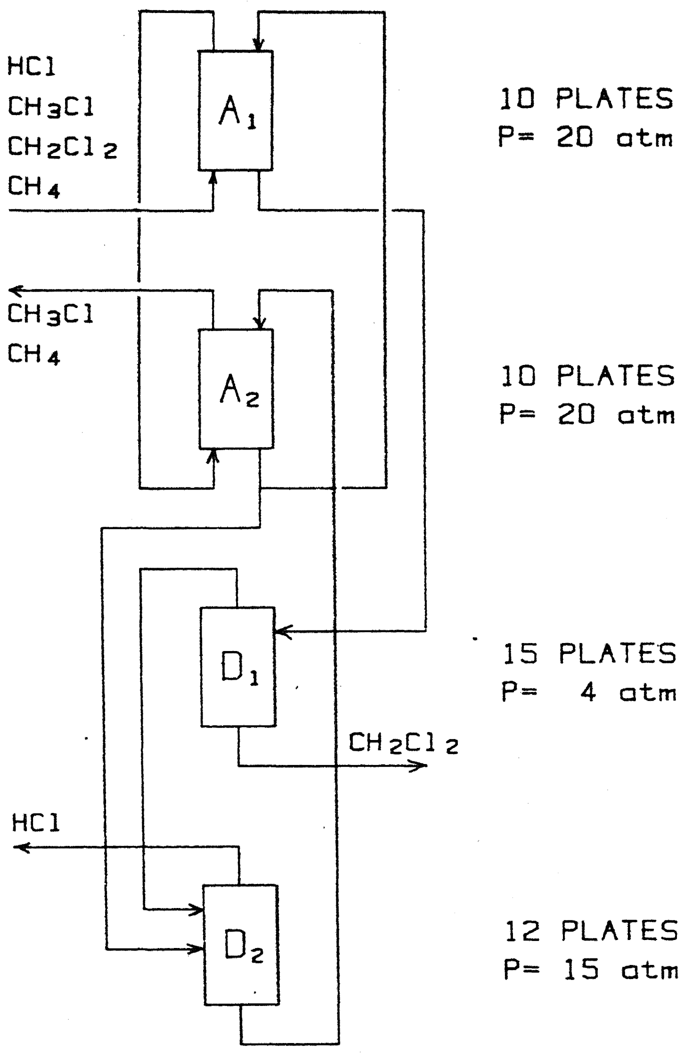


FIG 5 THE ARRANGEMENT OF THE INTERLINKED-COLUMNS (PROBLEM 4)

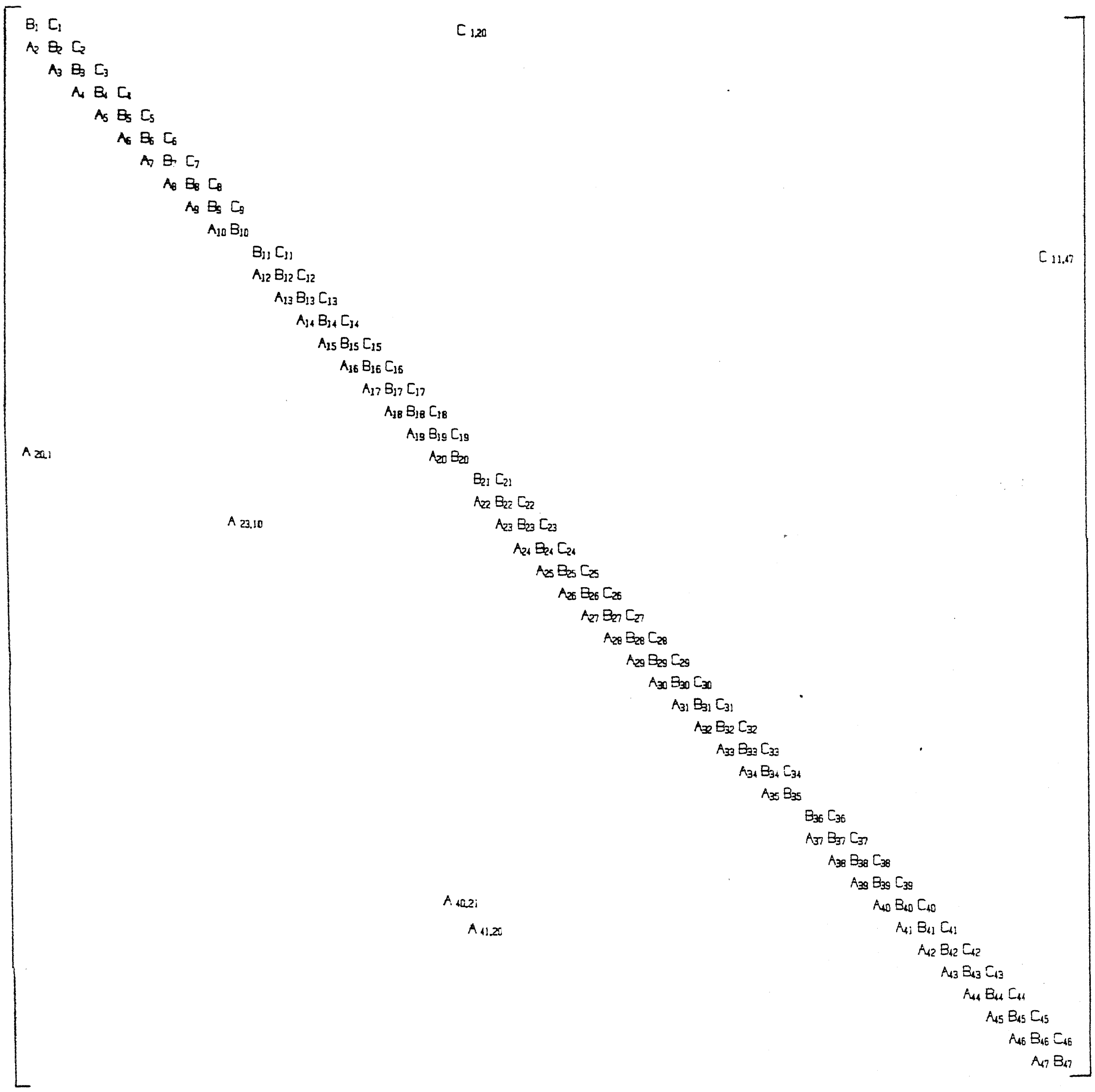


FIG 6 THE STRUCTURE OF THE JACOBIAN MATRIX
FOR PROBLEM 4

TABLE 4

Comparison of Operation Counts for Problem 4

Number of components = 4

Total number of stages= 47

Operations	Number of times performed	Standard operation count	Improved operation count
Op 1	42	x729 = 30618	x225 = 9450
Op 2	2	x729 = 1458	x 81 = 162
Op 3	43	x729 = 31347	x 41 = 1763
Op 4	38	x729 = 27702	x 41 = 1558
Op 6	2	x729 = 1458	x125 = 250
Op 8	36	x729 = 26244	x225 = 8100
Op 10	1	x729 = 729	x405 = 405
Op 11	25	x729 = 18225	x225 = 5625
Op 12	1	x729 = 729	x225 = 225
Op 15	45	x 81 = 3645	x 9 = 405
Op 16	2	x 81 = 162	x 25 = 50
Op 17	66	x 81 = 5346	x 45 = 2970
Op 18	40	x 81 = 3240	x 45 = 1800
Op 19	47	x 81 = 3807	x 81 = 3807
Op 20	47	x729 = 34263	x546 = 25662
Standard matrix multiplication	1	x729 = 729	x729 = 729
Total		189702	62961

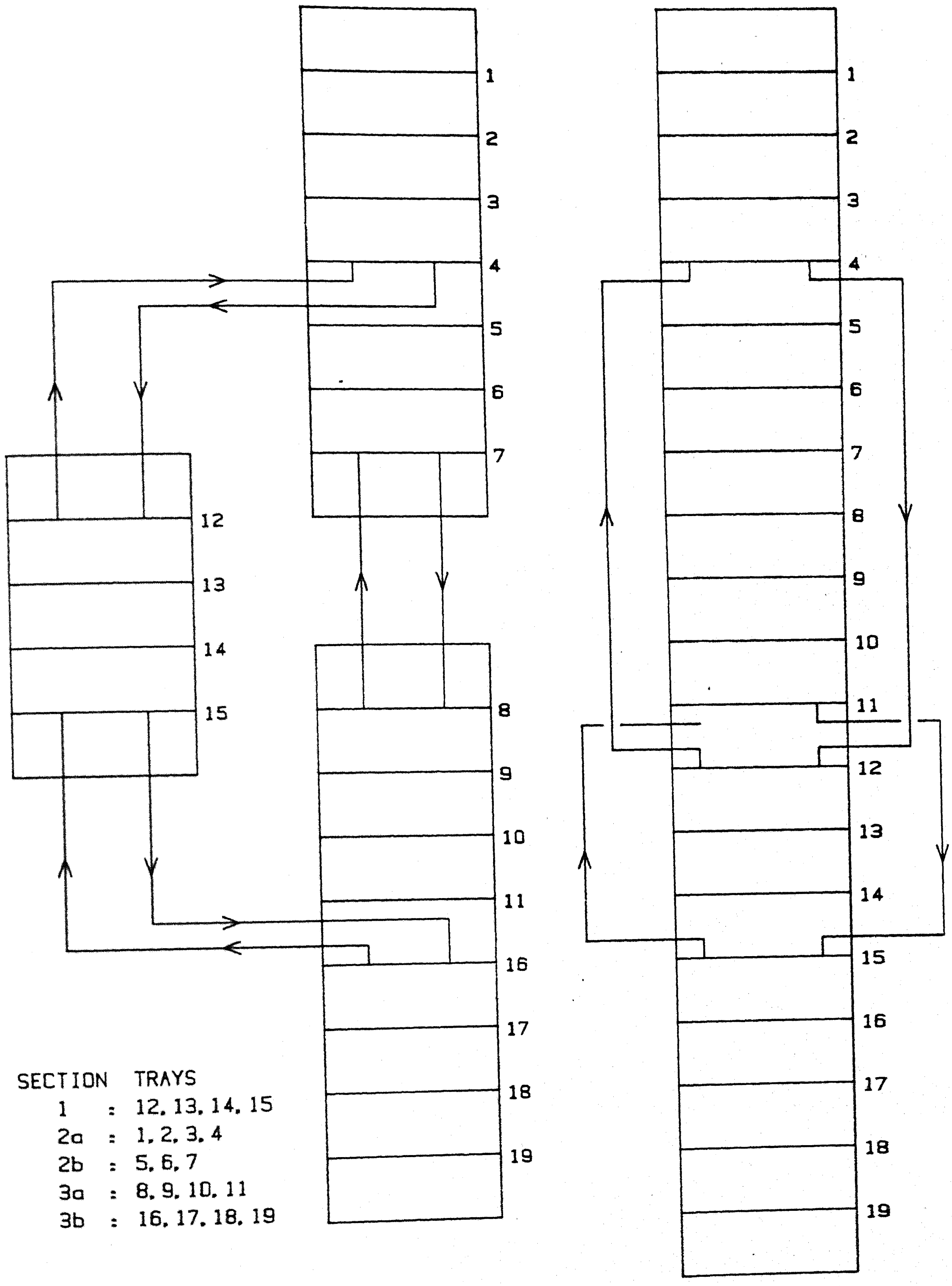
$$\frac{\text{S.O.C.}}{\text{I.O.C.}} = 3.013$$

It may be noted here that out of the 99 P-blocks, 15 are null matrices ($P_{11,20}$ through $P_{18,20}$, $P_{21,47}$, $P_{22,47}$, and $P_{35,47}$ through $P_{39,47}$) and hence need not be stored, thereby leaving only 85 P-blocks which must be stored. Therefore, only 3825 (= 85x9x5) elements are stored instead of 8019 (= 99x9x9), and a 52% of actual saving in the storage is obtained.

Problem 5: Consider the system of interlinked columns for separating a mixture of 3-components using 3 columns which are connected as shown in the Figure 7. These columns are assumed to be split into section 2a having 4 plates, 2b having 3 plates, 3a having 4 plates, 1 having 4 plates, and 3b having 4 plates, and are rearranged in that order. The Jacobian matrix for this system is presented in the Figure 8. The modified Thomas algorithm was applied to obtain the solution.

In this case, the sparsity exploitation has resulted in a reduction in the computations by a ratio of 2.744, as only 13393 operations were performed instead of the 36750. The comparison and details of the operation counts are presented in the Table 5.

There are 18 upper diagonal, 11 off-tridiagonal, and hence a total of 29 P-blocks, each of order 7x7, in the Jacobian matrix, (P_1 through P_{18}, $P_{4,12}$ through $P_{4,10}$, and $P_{11,16}$ through $P_{14,16}$) as shown in the Figure 5. Since only



SECTION		TRAYS
1	:	12, 13, 14, 15
2a	:	1, 2, 3, 4
2b	:	5, 6, 7
3a	:	8, 9, 10, 11
3b	:	16, 17, 18, 19

FIG 7 AN ARRANGEMENT OF INTERLINKED-COLUMNS
(PROBLEM 5 AND 6)

$$
\left[\begin{array}{ccccccccccccccccccc}
B_1 & C_1 & & & & & & & & & & & & & & & & & \\
A_2 & B_2 & C_2 & & & & & & & & & & & & & & & & \\
 & A_3 & B_3 & C_3 & & & & & & & & & & & & & & & \\
 & & A_4 & B_4 & C_4 & & & & & & & C_{4,12} & & & & & & & \\
 & & & A_5 & B_5 & C_5 & & & & & & & & & & & & & \\
 & & & & A_6 & B_6 & C_6 & & & & & & & & & & & & \\
 & & & & & A_7 & B_7 & C_7 & & & & & & & & & & & \\
 & & & & & & A_8 & B_8 & C_8 & & & & & & & & & & \\
 & & & & & & & A_9 & B_9 & C_9 & & & & & & & & & \\
 & & & & & & & & A_{10} & B_{10} & C_{10} & & & & & & & & \\
 & & & & & & & & & A_{11} & B_{11} & & & & & C_{11,16} & & & \\
 & & & A_{12,4} & & & & & & & & B_{12} & C_{12} & & & & & & \\
 & & & & & & & & & & & A_{13} & B_{13} & C_{13} & & & & & \\
 & & & & & & & & & & & & A_{14} & B_{14} & C_{14} & & & & \\
 & & & & & & & & & & & & & A_{15} & B_{15} & C_{15} & & & \\
 & & & & & & & & & & A_{16,11} & & & & A_{16} & B_{16} & C_{16} & & \\
 & & & & & & & & & & & & & & & A_{17} & B_{17} & C_{17} & \\
 & & & & & & & & & & & & & & & & A_{18} & B_{18} & C_{18} \\
 & & & & & & & & & & & & & & & & & A_{19} & B_{19}
\end{array}\right]
$$

FIG 8 THE STRUCTURE OF THE JACOBIAN MATRIX
(PROBLEM 5 AND 6)

TABLE 5

Comparison of Operation Counts for Problem 5

Number of components = 3

Total Number of stages = 19

Operation	Number of times performed	Standard operation count	Improved operation count
Op 1	18	x343 = 6174	x112 = 2016
Op 3	30	x343 = 10290	x25 = 750
Op 7	10	x343 = 3430	x112 = 1120
Op 9	1	x343 = 343	x196 = 196
Op 14	18	x343 = 6174	x112 = 2016
Op 15	19	x 49 = 931	x 7 = 133
Op 17	40	x 49 = 1960	x 28 = 1120
Op 19	19	x 49 = 931	x 49 = 931
Op 20	19	x343 = 6517	x269 = 5111
Total		36750	13393

$$\frac{\text{S.O.C.}}{\text{I.O.C.}} = 2.744$$

a total of 812 (= 29 x 7 x 4) elements are stored instead of the 1421 (= 29x7x7), a saving of about 43% in the storage requirement is obtained.

Problem 6: The same system of the interlinked columns described in the problem 2 was solved with a 6-component mixture. The structure of the Jacobian and the method of solution remains the same, but the order of each submatrix in the Jacobian is 13x13 now.

In this case, the sparsity exploitation has resulted in a reduction in the computations by a ratio of 3.21, which is more significant than that in the problem 2. Only 69811 operations were performed instead of the 224094, and the details are presented in the Table 6.

The total number of P-blocks is 29 (same as in problem 2). A total of only 2693 (= 29x13x7) elements are stored instead of 4901 (= 29x13x7), and thereby a saving of 46% can be realized in the storage requirements.

TABLE 6

Comparison of the Operation Counts for Problem 6

Number of components = 6

Total number of stages = 19

Operation	Number of times performed	Standard operation count	Improved operation count
Op 1	18	x2197 = 39546	x637 = 11466
Op 3	30	x2197 = 65910	x 85 = 2550
Op 7	10	x2197 = 21970	x637 = 6370
Op 9	1	x2197 = 2197	x1183= 1183
Op 14	18	x2197 = 39546	x637 = 11466
Op 15	19	x 169 = 3211	x 13 = 247
Op 17	40	x 169 = 6760	x 91 = 3640
Op 19	19	x 169 = 3211	x169 = 3211
Op 20	19	x2197 = 41743	x1562= 29678
		224094	69811

$$\frac{\text{S.O.C.}}{\text{I.O.C.}} = 3.21$$

CHAPTER 7

CONCLUSIONS

In the algorithm presented in this work the sparsity and the structure of the submatrices was exploited in the various matrix multiplications and inversions, and thereby a significant reduction in the operation-count and storage requirements was obtained. The reduction in the computations becomes increasingly significant with the increase in the number of components.

Theoretically, a maximum of 3.43 (= 24/7) times reduction in the computations may be realized in the single column problems if the number of components is very large. In general, a reduction in storage requirement by a ratio of $(c+1)/(2c+1)$ was also realized. The exploitation of sparsity in solving the single column problems by the conventional block Thomas algorithm has shown 2.676 and 3.02 times improvement in the operation-count, in the test problems 1 and 2.

An efficient approach to solve the problems with the intermediate tray specifications has been proposed and shown to be more advantageous than the one proposed by Hofeling and Seader [2] , in both the computational and the storage aspects. For instance, in the problem 3, the proposed approach has been proved to be 3.877 times computationally efficient and 64% more storage-saving than the Hofeling-Seader approach.

In the solution of a system of interlinked columns by the modified Thomas algorithm, the exploitation of sparsity has resulted in a 3.013, 2.744 and 3.21 times improvement in the operation count, in the problems 4,5 and 6 respectively. Though this ratio varies from problem to problem, and improves with the increase in the number of components, in general it is close to 3.

The saving, at least by a ratio of $(c+1)/(2c+1)$ in the storage requirement is also realized in the systems of interlinked columns, which significantly improves further if some of the off-tridiagonal P-blocks are null matrices.

The proposed method of the sparsity exploitation can be applied to the other related techniques, for instance, the convergence domain extension methods proposed by Vickery and Taylor [8] ; and the other methods of solution proposed by Kubicek [7] , and Stadtherr [5].

REFERENCES

1. Naphtali, L.M., and Sandholm, D.P., AIChE Journal, 17, 1 (1971).

2. Hofeling, B.S., and Seader, J.D., AIChE Journal, 24, 1131 (1978).

3. Ketchum, R.G., Chemical Engineering Science, 34, 387 (1979).

4. Hildalgo, R.S., Correa, A.V., Gomez, A.M., Seader, J.D., AIChE Journal, 26, 585 (1980).

5. Stadtherr, M.A., and Malachowski, M.A., Comput. and Chem. Engg., 6, 121 (1982).

6. Seader, J.D., AIChE Journal, 17, 125 (1980).

7. Kubicek, M., Communications of the ACM, 16, 760 (1973).

8. Vickery, J.D., and Taylor, R., AIChE Journal, 32, 547 (1986).

9. Islam, S., M. Tech. Thesis, I.I.T. Kanpur (1985).

10. Holland, C.D., Fundamentals of Multicomponent Distillation, McGraw Hill, New York (1981).

11. Kubicek, M., Halavacek, V., Prochaka, F., Chemical Engineering Science, 31, 277 (1976).

12. Browne, D.W., Ishii, Y., Otto, F.D., Canadian Journal of Chemical Engineering, 55, 307 (1977).

APPENDIX

ELEMENTS OF THE SUBMATRICES

TRIDIAGONAL SUBMATRICES:

(i) For Matrix A

$$\frac{\partial M_{j,i}}{\partial l_{j-1,k}} = r_{Lj-1}\,\delta_{ik}\,; \quad \frac{\partial M_{j,i}}{\partial v_{j-1,k}} = 0; \quad \frac{\partial M_{j,i}}{\partial T_{j-1}} = 0$$

$$\frac{\partial Q_{j,i}}{\partial l_{j-1,k}} = \frac{\partial Q_{j,i}}{\partial v_{j-1,k}} = \frac{\partial Q_{j,i}}{\partial T_{j-1}} = 0$$

$$\frac{\partial E_j}{\partial l_{j-1,i}} = r_{Lj-1} h_{j-1,i}\,; \quad \frac{\partial E_j}{\partial v_{j-1,i}} = 0; \quad \frac{\partial E_j}{\partial T_{j-1}} = r_{Lj-1}$$

$$\sum_{i=1}^{c} l_{j-1,i} \frac{\partial h_{j-1,i}}{\partial T_{j-1}}$$

(ii) For Matrix B

$$\frac{\partial M_{j,i}}{\partial l_{j,k}} = -(1+s_j)\delta_{i,k}; \quad \frac{\partial M_{j,i}}{\partial v_{j,k}} = -(1+S_j)\delta_{i,k}; \quad \frac{\partial M_{j,i}}{\partial T_j} = 0$$

$$\frac{\partial Q_{j,i}}{\partial l_{j,k}} = \frac{\eta_j}{L_j^2}\Big[K_{j,i}\,\delta_{i,k}\,L_j + l_{j,i} \sum_{p=1}^{c} \Big(\frac{\partial K_{j,i}}{\partial x_{j,p}}\Big)$$

$$\Big(\delta_{k,p} - \frac{l_{j,p}}{L_j}\Big) - K_{j,i} l_{j,i} \Big]$$

$$\frac{\partial Q_{j,i}}{\partial v_{j,k}} = \frac{1}{V_j}\left[\frac{\eta_j\, l_{j,i}}{L_j}\sum_{p=1}^{c}\left(\frac{\partial K_{j,i}}{\partial y_{j,p}}\right)\left(\delta_{p,k} - \frac{v_{j,p}}{V_j}\right) - \delta_{i,k} + \frac{v_{j,i}}{V_j}\right]$$

$$\frac{\partial Q_{j,i}}{\partial T_j} = \frac{\eta_j\, l_{j,i}}{L_j}\left(\frac{\partial K_{j,i}}{\partial T_j}\right)$$

$$\frac{\partial E_j}{\partial l_{j,i}} = -(1 + s_j)h_{j,i}\ ; \quad \frac{\partial E_j}{\partial v_{j,i}} = -(1 + S_j)H_{j,i}$$

$$\frac{\partial E_j}{\partial T_j} = -(1 + s_j)\sum_{i=1}^{c} l_{j,i}\frac{\partial h_{j,i}}{\partial T_j} - (1+S_j)\sum_{i=1}^{c} v_{j,i}\frac{\partial H_{j,i}}{\partial T_j}$$

(iii) For Matrix C

$$\frac{\partial M_{j,i}}{\partial l_{j+1,k}} = 0; \quad \frac{\partial M_{j,i}}{\partial v_{j+1,k}} = r_{Vj+1}\delta_{i,k}\ ; \quad \frac{\partial M_{j,i}}{\partial T_{j+1}} = 0$$

$$\frac{\partial Q_{j,i}}{\partial l_{j+1,k}} = 0; \quad \frac{\partial Q_{j,i}}{\partial T_{j+1}} = 0$$

$$\frac{\partial Q_{j,i}}{\partial v_{j+1,k}} = \frac{(1-\eta_j)r_{Vj+1}}{D}\left[\delta_{i,k} - \frac{r_{Vj+1}v_{j+1,i}+R_{jy}v_{y,i}+R_{jz}v_{zi}}{D}\right]$$

where,

$$D = r_{Vj+1} V_{j+1} + R_{jy} V_y + R_{jz} V_z$$

$$\frac{\partial E_j}{\partial l_{j+1,k}} = 0; \quad \frac{\partial E_j}{\partial v_{j+1,i}} = r_{Vj+1} H_{j+1,i}$$

$$\frac{\partial E_j}{\partial T_{j+1}} = r_{Vj+1} \sum_{i=1}^{c} v_{j+1,i} \left(\frac{\partial H_{j+1,i}}{\partial T_{j+1}}\right)$$

OFFDIAGONAL SUBMATRICES:

(i) For A-matrix:

$$\frac{\partial M_{j,i}}{\partial l_{p,k}} = R_{j,p}\,\delta_{i,k}\ ; \quad \frac{\partial M_{j,i}}{\partial v_{p,k}} = \frac{\partial M_{j,i}}{\partial T_p} = 0$$

$$\frac{\partial Q_{j,i}}{\partial l_{p,k}} = \frac{\partial Q_{j,i}}{\partial v_{p,k}} = \frac{\partial Q_{j,i}}{\partial T_p} = 0$$

$$\frac{\partial E_j}{\partial l_{p,k}} = R_{j,p} h_{p,i}\ ; \quad \frac{\partial E_j}{\partial v_{p,k}} = 0; \quad \frac{\partial E_j}{\partial T_p} = R_{j,p} \sum_{i=1}^{c} l_{p,i} \frac{\partial h_{p,i}}{\partial T_p}$$

(ii) For C-matrix:

$$\frac{\partial M_{j,i}}{\partial l_{y,k}} = 0; \quad \frac{\partial M_{j,i}}{\partial v_{y,k}} = R_{j,y}\delta_{i,k}\ ; \quad \frac{\partial M_{j,i}}{\partial T_y} = 0$$

$$\frac{\partial Q_{j,i}}{\partial l_{y,k}} = \frac{\partial Q_{j,i}}{\partial T_y} = 0$$

$$\frac{\partial Q_{j,i}}{\partial v_{j,k}} = \frac{(1-\eta_j)R_{j,y}}{D} \left[\delta_{i,k} - \frac{r_{Vj+1} v_{j+1,i} + R_{j,y} v_{y,i} + R_{j,z} v_{z,i}}{D}\right]$$

where

$$D = r_{Vj+1} V_{j+1} + R_{j,y} v_{y,i} + R_{j,z} v_{z,i}$$

$$\frac{\partial E_j}{\partial l_{y,i}} = 0; \quad \frac{\partial E_j}{\partial v_{y,i}} = R_{j,y} H_{y,i}; \quad \frac{\partial E_j}{\partial T_y} = R_{j,y} \sum_{i=1}^{c} v_{y,i} \left(\frac{\partial H_{y,i}}{\partial T_y}\right).$$

APPENDIX B1 (SAMPLE PROBLEM NO. 1)

K values for the temperature range of - 25°F to 40°F and at a pressure of 800 lb/in^{2}abs

Component	a_{11}	a_{21}	a_{31}	a_{41}
CO_2	-0.62822223×10^{-1}	0.30688802×10^{-3}	0.39996468×10^{-6}	-0.57899830×10^{-9}
N_2	0.50596821	-0.43488364×10^{-3}	-0.15009991×10^{-5}	0.34494154×10^{-8}
CH_4	0.15584934	-0.15205775×10^{-3}	0.50349212×10^{-6}	-0.17713546×10^{-9}
C_2H_6	0.91486037×10^{-1}	-0.16355944×10^{-3}	0.33741924×10^{-6}	0.14797150×10^{-9}
C_3H_8	0.37769508×10^{-1}	-0.64491702×10^{-4}	0.29233627×10^{-6}	$-0.48597680\times10^{-11}$
i C_4H_{10}	0.36708355×10^{-1}	-0.94310963×10^{-4}	0.28026648×10^{-6}	0.10462797×10^{-10}
n C_4H_{10}	0.37231278×10^{-1}	-0.13635085×10^{-3}	0.37584653×10^{-6}	$-0.69237741\times10^{-10}$
n C_5H_{12}	0.15414596×10^{-1}	-0.34736106×10^{-4}	0.12591028×10^{-6}	0.73157133×10^{-10}
i C_5H_{12}	0.19747034×10^{-1}	-0.40284984×10^{-4}	0.14439195×10^{-6}	0.56656790×10^{-10}
n C_6H_{14}	0.88765752×10^{-3}	0.37082646×10^{-4}	-0.40746951×10^{-7}	0.15187203×10^{-9}
n C_7H_{16}	0.63677356×10^{-2}	-0.64409760×10^{-5}	0.31793974×10^{-7}	0.78284379×10^{-10}
n C_8H_{18}	0.99674799×10^{-2}	-0.34673591×10^{-4}	0.82305291×10^{-7}	0.21022392×10^{-10}
n C_9H_{20}	0.78793392×10^{-2}	-0.23886125×10^{-4}	0.62435951×10^{-7}	0.25793478×10^{-10}
n $C_{10}H_{22}$	0.64146556×10^{-2}	-0.16131104×10^{-4}	0.30005250×10^{-7}	0.30266026×10^{-10}

$K_1 = T\,(a_{11} + a_{21}\,T + a_{31}\,T^2 + a_{41}\,T^3)^3$ (T in °R)

Appendix B/ continued.

LIQUID ENTHALPIES FOR THE TEMPERATURE RANGE OF - 25°F to 40°F at P = 800 lb/in^{2}abs.

Component	b_{1i}	b_{2i}	b_{3i}	b_{4i}
CO_2	0.22524075×10^{4}	0.5446243×10^{1}	0.2791080×10^{-1}	-0.18765335×10^{-4}
N_2	0.15837112×10^{4}	0.3731512×10^{1}	0.17655857×10^{-1}	-0.14662071×10^{-4}
CH_4	0.81635181×10^{3}	0.7206460×10^{1}	0.15354034×10^{-1}	-0.84406456×10^{-5}
C_2H_6	0.97404712×10^{3}	0.11454294×10^{2}	0.79399594×10^{-2}	-0.42183183×10^{-6}
C_3H_8	0.21237510×10^{4}	0.46383524×10^{1}	0.31726830×10^{-1}	-0.12580301×10^{-4}
i C_4H_{10}	0.17543628×10^{4}	0.92456856×10^{1}	0.30206113×10^{-1}	-0.89584664×10^{-5}
n C_4H_{10}	0.32309192×10^{4}	0.66175545×10^{1}	0.38262386×10^{-1}	-0.16110935×10^{-4}
i C_5H_{12}	0.33611663×10^{4}	0.39552670×10^{1}	0.54925641×10^{-1}	-0.25869682×10^{-4}
n C_5H_{12}	0.43454375×10^{4}	0.10596339×10^{2}	0.43731511×10^{-1}	-0.19637475×10^{-4}
n C_6H_{14}	-0.44150469×10^{4}	0.70354599×10^{2}	-0.67470074×10^{-1}	0.60245657×10^{-4}
n C_7H_{16}	0.66707016×10^{2}	0.18159073×10^{2}	0.38164884×10^{-1}	-0.42837073×10^{-5}
n C_8H_{18}	-0.10632578×10^{2}	0.19229950×10^{2}	0.40186413×10^{-1}	-0.70521889×10^{-6}
n C_9H_{20}	-0.79141992×10^{4}	0.81615143×10^{2}	-0.79501927×10^{-1}	0.83943509×10^{-4}
n $C_{10}H_{22}$	-0.67810352×10^{4}	0.74108551×10^{2}	-0.58315706×10^{-1}	0.75087155×10^{-4}

$h_i = b_{1i} + b_{2i}T + b_{3i}T^2 + b_{4i}T^3$ (T in °R) Btu/lb mole.

Appendix B1 continued.

VAPOR ENTHALPIES FOR THE TEMPERATURE RANGE OF - 25°F To 40°F AT P = 800 lb/in^{2}abs.

Component	C_{11}	C_{21}	C_{31}	C_{41}
CO_2	$0.13978977x10^5$	$-0.96359463x10^1$	$0.38228422x10^{-1}$	$-0.26870170x10^{-4}$
N_2	$0.48638672x10^4$	$-0.21227379x10^1$	$0.17565668x10^{-1}$	$-0.11367006x10^{-4}$
CH_4	$0.63255430x10^4$	$-0.20747757x10^1$	$0.18532634x10^{-1}$	$-0.10630416x10^{-4}$
C_2H_6	$0.10628934x10^5$	$-0.28718834x10^1$	$0.24877094x10^{-1}$	$-0.13233222x10^{-4}$
C_3H_8	$0.13954383x10^5$	$-0.41930256x10^1$	$0.32614145x10^{-1}$	$-0.15483340x10^{-4}$
i C_4H_{10}	$0.94088984x10^4$	$0.39262680x10^2$	$-0.55596594x10^{-1}$	$0.51507392x10^{-4}$
n C_4H_{10}	$0.57302344x10^4$	$0.75117737x10^2$	$-0.13120884x10^0$	$0.10517908x10^{-3}$
i C_5H_{12}	$0.83081953x10^4$	$0.75267792x10^2$	$-0.12945843x10^0$	$0.10845697x10^{-3}$
n C_5H_{12}	$0.12804211x10^5$	$0.61654007x10^2$	$-0.97365201x10^{-1}$	$0.84398722x10^{-4}$
n C_6H_{14}	$0.23001684x10^5$	$0.27744919x10^2$	$-0.31545494x10^{-1}$	$0.49981289x10^{-4}$
n C_7H_{16}	$0.14876816x10^5$	$0.59342438x10^2$	$-0.81853271x10^{-1}$	$0.81429855x10^{-4}$
n C_8H_{18}	$0.32793215x10^5$	$-0.35040283x10^2$	$0.11162955x10^0$	$-0.42647429x10^{-4}$
n C_9H_{20}	$0.47024656x10^5$	$-0.95395035x10^2$	$0.24547529x10^0$	$-0.13209638x10^{-3}$
n $C_{10}H_{22}$	$0.55238211x10^5$	$-0.13195618x10^3$	$0.32518369x10^0$	$-0.18188384x10^{-3}$

$H_1 = C_{11} + C_{21}\ T + C_{31}\ T^2 + C_{41}\ T^3$ (T in °R) Btu/lb mole.

APPENDIX B2

SAMPLE PROBLEM No.2

DATA

K values

Material	Temperature	
	100°F	200°F
A	500.0	550.0
B	1.50	1.8
C	0.90	1.00
D	1.0×10^{-6}	1.5×10^{-6}
	Molar liquid enthalpies, 10^3 cal/mole.	
A	0.01	0.013
B	0.30	0.33
C	0.40	0.44
D	1.50	1.90
	Molar vapor enthalpies, 10^3 cal/mole	
A	1.00	1.002
B	1.80	1.82
C	2.00	2.03
D	5.75	5.95

```
!---------------------------------------------------------------
! INTERLINKED COLUMNS
!---------------------------------------------------------------
        PROGRAM INTLNK
        INTEGER C,CP1,TWOC,CT
        DIMENSION JCOND(2),ITYPEC(2),JREBL(2),ITYPER(2),
        1 IX(6),IY(6),LORV(6),RATIO(6),JNOSV(3),JNOSL(3),
        1 JVAPF(1),TFV(1),FEEDV(1,1),JLIQF(1),TFL(1),FEEDL(1,1),
        1 JSPC(4),LBL(4),SPCVAL(4),ICOMP(4),
        1 QQ(12),SV(12,2),SL(12,2),SSV(12),SSL(12),T(12),ETA(12),
        1 HV(4),HL(4),AK(4),VECTOR(4),
        1 BV(47),BL(47),RV(47),RL(47),DENOM(47)
        DIMENSION A(9,9),B(9,9),CC(9,9),AOFF(9,9),COFF(9,9)
        DIMENSION ALFA(9,5),BETA2(9,5),BETA1(9),VEC1(9)
        DIMENSION F(47,9),P(85,9,5),Q(47,9)
!---------------------------------------------------------------
        OPEN(UNIT=33,DEVICE='DSK',FILE='INP.IN')
        OPEN(UNIT=51,DEVICE='DSK',FILE='TH.OUT')
        READ(33,*)N
        READ(33,*)C,CP1,TWOC,CT
        READ(33,*)NOFF,NABOVE
        READ(33,*)NCOND,NREBL,NOV,NOL,NVAPF,NLIQF,NSPC
        IF(NCOND.LT.1)GOTO 810
        DO 800 I=1,NCOND
800     READ(33,*)JCOND(I),ITYPEC(I)
810     IF(NREBL.LT.1)GOTO 811
        DO 801 I=1,NREBL
801     READ(33,*)JREBL(I),ITYPER(I)
811     IF(NSPC.LT.1)GOTO 812
        DO 802 I=1,NSPC
802     READ(33,*)JSPC(I),LBL(I),SPCVAL(I),ICOMP(I)
812     IF(NOV.LT.1)GOTO 813
        READ(33,*)(JNOSV(I),I=1,NOV)
813     IF(NOL.LT.1)GOTO 814
        READ(33,*)(JNOSL(I),I=1,NOL)
814     IF(NOFF.LT.1)GOTO 815
        DO 803 I=1,NOFF
803     READ(33,*)IX(I),IY(I),LORV(I),RATIO(I)
815     IF(NVAPF.LT.1)GOTO 816
        DO 804 I=1,NVAPF
804     READ(33,*)JVAPF(I),TFV(I),(FEEDV(I,KK),KK=1,C)
816     IF(NLIQF.LT.1)GOTO 817
        DO 805 I=1,NLIQF
805     READ(33,*)JLIQF(I),TFL(I),(FEEDL(I,KK),KK=1,C)
817     CONTINUE
        READ(33,*)(QQ(J),J=1,N)
        READ(33,*)(SSV(J),J=1,N)
        READ(33,*)(SSL(J),J=1,N)
        READ(33,*)(ETA(J),J=1,N)
        READ(33,*)(T(J),J=1,N)
        READ(33,*)((SV(J,I),I=1,C),J=1,N)
        READ(33,*)((SL(J,I),I=1,C),J=1,N)
!---------------------------------------------------------------
        ICHK=0
        KCOND=0
        KREBL=0
        NTOT=NCOND+NREBL
        DO 110 KK=1,NSPC
        IF(LBL(KK).LT.3)GOTO 110
        J=JSPC(KK)
        DO 111 MM=1,NCOND
        IF(J.EQ.JCOND(MM))GOTO 113
111     CONTINUE
        DO 112 MM=1,NREBL
        IF(J.EQ.JREBL(MM))GOTO 114
112     CONTINUE
        GOTO 115
113     KCOND=KCOND+1
```

```
        GOTO 115
114     KREBL=KREBL+1
115     ICHK=ICHK+1
110     CONTINUE
        ISTP=0
        IF(KCOND.EQ.NCOND)GOTO 116
        ISTP=1
        TYPE 901,KCOND,NCOND
116     IF(KREBL.EQ.NREBL)GOTO 117
        ISTP=1
        TYPE 902,KREBL,NREBL
117     IF(ICHK.EQ.NTOT)GOTO 119
        IF(ICHK.LT.NTOT)GOTO 118
        TYPE 904,ICHK,NTOT
        STOP
118     TYPE 903,ICHK,NTOT
        STOP
901     FORMAT(1X,'---ERROR---'/1X,'INFORMATION ABOUT ',I2,' CONDENSERS'
       1 ,' HAS BEEN PROVIDED WHEREAS THERE ARE ',I2,' CONDENSERS')
902     FORMAT(1X,'---ERROR---'/1X,'INFORMATION ABOUT ',I2,' REBOILERS'
       1 ,' HAS BEEN PROVIDED WHEREAS THERE ARE ',I2,' REBOILERS')
903     FORMAT(1X,'---ERROR---'/1X,'NO. OF SPECIFICATIONS SHOULD BE ',I2
       1 ,' BUT ',I2,' HAVE BEEN PROVIDED'/1X,'PROBLEM UNDERSPECIFIED')
904     FORMAT(1X,'---ERROR---'/1X,'NO. OF SPECIFICATIONS SHOULD BE ',I2
       1 ,' BUT ',I2,' HAVE BEEN PROVIDED'/1X,'PROBLEM OVERSPECIFIED')
119     IF(ISTP.EQ.1)STOP
!-----------------------------------------------------------------
        DO 145 J=1,N
        RL(J)=1.
145     RV(J)=1.
        DO 146 KK=1,NOFF
        J=IY(KK)
        IF(LORV(KK).EQ.0)GOTO 147
        RV(J)=RV(J)-RATIO(KK)
        GOTO 146
147     RL(J)=RL(J)-RATIO(KK)
146     CONTINUE
        DO 151 J=1,N
        BV(J)=0.
        BL(J)=0.
        DO 151 I=1,C
        BV(J)=BV(J)+SV(J,I)
151     BL(J)=BL(J)+SL(J,I)
        DO 200 J=1,N
        JM1=J-1
        JP1=J+1
        TJ=T(J)
        SD0=-(1.+SSL(J))
        SD1=-(1.+SSV(J))
        DO 152 KK=1,NCOND
        IF(J.EQ.JCOND(KK))GOTO 180
152     CONTINUE
        DO 153 KK=1,NREBL
        IF(J.EQ.JREBL(KK))GOTO 190
153     CONTINUE
        CALL ENV(TJ,HV)
        CALL ENL(TJ,HL)
        CALL FINDK(TJ,AK)
        SUMV=0.
        SUML=0.
        COEF=ETA(J)*BV(J)/BL(J)
        DO 154 I=1,C
        F(J,I)=SD0*SL(J,I)+SD1*SV(J,I)
        F(J,I+C)=COEF*AK(I)*SL(J,I)-SV(J,I)
        SUMV=SUMV+HV(I)*SV(J,I)
154     SUML=SUML+HL(I)*SL(J,I)
        F(J,CT)=SD0*SUML+SD1*SUMV+QQ(J)
        IF(J.EQ.1)GOTO 170
        DO 155 KK=1,NOL
        IF(J.EQ.JNOSL(KK))GOTO 170
155     CONTINUE
        CALL ENL(T(J-1),HL)
        SUMHL=0.0
        DO 157 I=1,C
        F(J,I)=F(J,I)+SL(JM1,I)*RL(JM1)
157     SUMHL=SUMHL+HL(I)*SL(JM1,I)
        F(J,CT)=F(J,CT)+SUMHL*RL(JM1)
170     IF(J.EQ.N)GOTO 200
        DO 171 KK=1,NOV
        IF(J.EQ.JNOSV(KK))GOTO 200
171     CONTINUE
        CALL ENV(T(J+1),HV)
        SUMHV=0.
        DO 173 I=1,C
        F(J,I)=F(J,I)+SV(JP1,I)*RV(JP1)
173     SUMHV=SUMHV+HV(I)*SV(JP1,I)
        F(J,CT)=F(J,CT)+SUMHV*RV(JP1)
        GOTO 200
!-------------------------------------------------------------------
! CONDENSER PART
180     DO 181 I=1,C
181     F(J,I)=SV(JP1,I)*RV(JP1)+SD1*SV(J,I)+SD0*SL(J,I)
        IF(ITYPEC(KK).EQ.0)GOTO 182
        DO 184 I=1,C
184     F(J,I+C)=BV(J)*SL(J,I)/BL(J)-SV(J,I)
        GOTO 200
182     CALL FINDK(TJ,AK)
        COEF1=ETA(J)*BV(J)/BL(J)
        COEF2=(1.-ETA(J))*BV(J)/BV(JP1)
        DO 183 I=1,C
183     F(J,I+C)=COEF1*AK(I)*SL(J,I)-SV(J,I)+COEF2*SV(JP1,I)
        GOTO 200
!-------------------------------------------------------------------
! REBOILER PART
190     DO 191 I=1,C
191     F(J,I)=SL(JM1,I)*RL(JM1)+SD1*SV(J,I)+SD0*SL(J,I)
        IF(ITYPER(KK).EQ.0)GOTO 192
        DO 194 I=1,C
194     F(J,I+C)=BV(J)*SL(J,I)/BL(J)-SV(J,I)
        GOTO 200
192     CALL FINDK(TJ,AK)
        COEF=BV(J)/BL(J)
        DO 193 I=1,C
193     F(J,I+C)=COEF*AK(I)*SL(J,I)-SV(J,I)
200     CONTINUE
!-------------------------------------------------------------------
! FEED PART OF THE DISCREPENCY FUNCTIONS
!-------------------------------------------------------------------
        DO 201 KK=1,NLIQF
        J=JLIQF(KK)
        TF=TFL(KK)
        CALL ENL(TF,HL)
        DO 201 I=1,C
        F(J,I)=F(J,I)+FEEDL(KK,I)
201     F(J,CT)=F(J,CT)+HL(I)*FEEDL(KK,I)
        DO 202 KK=1,NVAPF
        J=JVAPF(KK)
        TF=TFV(KK)
        CALL ENV(TF,HV)
        DO 202 I=1,C
        F(J,I)=F(J,I)+FEEDV(KK,I)
202     F(J,CT)=F(J,CT)+HV(I)*FEEDV(KK,I)
!-------------------------------------------------------------------
! CONTRIBUTION OF INTERLINKS TO THE DISCREPENCY FUNCTIONS
!-------------------------------------------------------------------
        DO 210 KK=1,NOFF
        J=IX(KK)
        JK=IY(KK)
        TK=T(JK)
        IF(LORV(KK).EQ.0)GOTO 211
        CALL ENV(TK,HV)
        SUMV=0.
        DO 213 I=1,C
        F(J,I)=F(J,I)+SV(JK,I)*RATIO(KK)
213     SUMV=SUMV+HV(I)*SV(JK,I)
        F(J,CT)=F(J,CT)+SUMV*RATIO(KK)
        GOTO 210
211     CALL ENL(TK,HL)
        SUML=0.
        DO 212 I=1,C
        F(J,I)=F(J,I)+SL(JK,I)*RATIO(KK)
212     SUML=SUML+SL(JK,I)*HL(I)
        F(J,CT)=F(J,CT)+SUML*RATIO(KK)
210     CONTINUE
!---------------------------------------------------------------
! LAST TERM OF THE EQUILIBRIUM DISCREPENCY FUNCTION
!---------------------------------------------------------------
        DO 240 J=1,N
        IF(ETA(J).EQ.1.)GOTO 240
        DO 220 KK=1,NCOND
        IF(J.EQ.JCOND(KK))GOTO 240
220     CONTINUE
        DO 221 KK=1,NREBL
        IF(J.EQ.JREBL(KK))GOTO 240
221     CONTINUE
        KK=1
222     IF(J.NE.IX(KK))GOTO 223
        IF(LORV(KK).EQ.1)GOTO 230
223     KK=KK+1
        IF(KK.LE.NOFF)GOTO 222
        DO 225 KK=1,NOV
        IF(J.EQ.JNOSV(KK))GOTO 226
225     CONTINUE
        COEFF=(1.-ETA(J))*BV(J)/BV(J+1)
        DO 227 I=1,C
227     F(J,I+C)=F(J,I+C)+COEFF*SV(J+1,I)
        DENOM(J)=RV(J+1)*BV(J+1)
        GOTO 240
226     TYPE 905,J
905     FORMAT(1X,'---ERROR---'/1X,'STAGE ',I2,' DOES NOT HAVE ANY ',
     1 'VAPOR INPUT AND HAS NOT BEEN DEFINED AS A REBOILER')
        STOP
230     LL=KK+1
        JK=IY(KK)
        DENOM(J)=BV(JK)*RATIO(KK)
        COEFF=(1.-ETA(J))*BV(J)
        DO 231 I=1,C
231     VECTOR(I)=SV(JK,I)*RATIO(KK)
        IF(LL.GT.NOFF)GOTO 232
        DO 232 KK=LL,NOFF
        IF(J.NE.IX(KK))GOTO 232
        IF(LORV(KK).EQ.0)GOTO 232
        JK=IY(KK)
        DENOM(J)=DENOM(J)+BV(JK)*RATIO(KK)
        DO 233 I=1,C
233     VECTOR(I)=VECTOR(I)+SV(JK,I)*RATIO(KK)
232     CONTINUE
        DO 235 KK=1,NOV
        IF(J.EQ.JNOSV(KK))GOTO 238
235     CONTINUE
        DENOM(J)=DENOM(J)+BV(J+1)*RV(J+1)
        DO 237 I=1,C
237     F(J,I+C)=F(J,I+C)+COEFF*(VECTOR(I)+SV(J+1,I)*RV(J+1))/DENOM(J)
        GOTO 240
```

```
238     DO 239 I=1,C
239     F(J,I+C)=F(J,I+C)+COEFF*VECTOR(I)/DENOM(J)
240     CONTINUE
!-----------------------------------------------------------------
! SPECIFICATIONS PART
!-----------------------------------------------------------------
        KK=1
270     J=JSPC(KK)
        JP1=J+1
        JM1=J-1
        TJ=T(J)
        SD0=-(1.+SSL(J))
        SD1=-(1.+SSV(J))
        IF(LBL(KK).LT.5)GOTO 260
        DO 251 MM=1,NCOND
        IF(J.EQ.JCOND(MM))GOTO 255
251     CONTINUE
        DO 252 MM=1,NREBL
        IF(J.EQ.JREBL(MM))GOTO 253
252     CONTINUE
        GOTO 260
253     QQ(J)=0.
        CALL ENL(T(J-1),HL)
        CALL ENV(TJ,HV)
        DO 254 I=1,C
254     QQ(J)=QQ(J)+(HL(I)-HV(I))*SV(J,I)
        QQ(J)=QQ(J)*SD1
        GOTO 260
255     QQ(J)=0.
        CALL ENV(T(J+1),HV)
        CALL ENL(TJ,HL)
        IF(ITYPEC(MM).EQ.0)GOTO 256
        DO 258 I=1,C
258     QQ(J)=QQ(J)+(HL(I)-HV(I))*SV(JP1,I)
        QQ(J)=QQ(J)*RV(J+1)
        GOTO 260
256     DO 257 I=1,C
257     QQ(J)=QQ(J)+(HV(I)-HL(I))*SL(J,I)
        QQ(J)=QQ(J)*SD0
260     GOTO(10,20,30,40,50,55,60,65,70,75,80,85,90)LBL(KK)
!-----------------------------------------------------------------
! NO CONDENSER SPECIFICATIONS
10      DO 11 MM=1,NCOND
        IF(J.EQ.JCOND(MM))GOTO 12
11      CONTINUE
        TYPE 906,J
906     FORMAT(1X,'---ERROR---'/1X,'STAGE ',I2,' IS NOT A CONDENSER')
        STOP
12      QQ(J)=0.
        F(J,CT)=0.
        CALL ENL(TJ,HL)
        CALL ENV(T(J+1),HV)
        IF(ITYPEC(MM).EQ.0)GOTO 14
        DO 13 I=1,C
        QQ(J)=QQ(J)+SV(JP1,I)*(HL(I)-HV(I))
13      F(J,CT)=F(J,CT)+HL(I)*(RV(JP1)*SV(JP1,I)+SD0*SL(J,I)
        1 +SD1*SV(J,I))
        QQ(J)=QQ(J)*RV(JP1)
        GOTO 265
14      DO 15 I=1,C
        QQ(J)=QQ(J)+(HV(I)-HL(I))*SL(J,I)
15      F(J,CT)=F(J,CT)+HV(I)*(RV(JP1)*SV(JP1,I)+SD0*SL(J,I))
        QQ(J)=QQ(J)*SD0
        CALL ENV(TJ,HV)
        SUM=0.
        DO 16 I=1,C
16      SUM=SUM+HV(I)*SV(J,I)
        F(J,CT)=F(J,CT)+SD1*SUM
        GOTO 265
!-------------------------------------------------------------------
! NO REBOILER SPECIFICATIONS
20      DO 21 MM=1,NREBL
        IF(J.EQ.JREBL(MM))GOTO 22
21      CONTINUE
        TYPE 907,J
907     FORMAT(1X,'---ERROR---'/1X,'STAGE ',I2,' IS NOT A REBOILER')
        STOP
22      QQ(J)=0.
        F(J,CT)=0.
        CALL ENL(T(J-1),HL)
        CALL ENV(TJ,HV)
        IF(ITYPER(MM).EQ.0)GOTO 23
        DO 26 I=1,C
        QQ(J)=QQ(J)+(HL(I)-HV(I))*SV(J,I)
26      F(J,CT)=F(J,CT)+HL(I)*(RL(JM1)*SL(JM1,I)+SD0*SL(J,I)+SD1
        1 *SV(J,I))
        QQ(J)=SD1*QQ(J)
        GOTO 265
23      DO 24 I=1,C
        QQ(J)=QQ(J)+(HL(I)-HV(I))*SV(J,I)
24      F(J,CT)=F(J,CT)+HL(I)*(RL(JM1)*SL(JM1,I)+SD1*SV(J,I))
        QQ(J)=SD1*QQ(J)
        CALL ENL(TJ,HL)
        SUM=0.
        DO 25 I=1,C
25      SUM=SUM+HL(I)*SL(J,I)
        F(J,CT)=F(J,CT)+SD0*SUM
        GOTO 265
!-------------------------------------------------------------------
! CONDENSER-HEAT-DUTY SPECIFIED
30      DO 31 MM=1,NCOND
        IF(J.EQ.JCOND(MM))GOTO 32
31      CONTINUE
        TYPE 908,J
908     FORMAT(1X,'---ERROR---'/1X,'STAGE ',I2,' IS NOT A CONDENSER')
        STOP
32      F(J,CT)=SPCVAL(KK)
        QQ(J)=SPCVAL(KK)
        CALL ENV(T(J+1),HV)
        CALL ENL(TJ,HL)
        SUMHV=0.
        DO 33 I=1,C
33      SUMHV=SUMHV+HV(I)*SV(JP1,I)
        F(J,CT)=F(J,CT)+RV(JP1)*SUMHV
        CALL ENV(TJ,HV)
        IF(ITYPEC(MM).EQ.0)GOTO 35
        DO 34 I=1,C
34      F(J,CT)=F(J,CT)+HL(I)*(SD0*SL(J,I)+SD1*SV(J,I))
        GOTO 265
35      SUML=0.
        SUMV=0.
        DO 36 I=1,C
        SUMV=SUMV+HV(I)*SV(J,I)
36      SUML=SUML+HL(I)*SL(J,I)
        F(J,CT)=F(J,CT)+SD0*SUML+SD1*SUMV
        GOTO 265
!-------------------------------------------------------------------
! REBOILER-HEAT-DUTY SPECIFIED
40      DO 41 MM=1,NREBL
        IF(J.EQ.JREBL(MM))GOTO 42
41      CONTINUE
        TYPE 909,J
909     FORMAT(1X,'---ERROR---'/1X,'STAGE ',I2,' IS NOT A REBOILER')
        STOP
42      F(J,CT)=SPCVAL(KK)
        QQ(J)=SPCVAL(KK)
        CALL ENL(T(J-1),HL)
        CALL ENV(TJ,HV)
        SUM=0.
        DO 43 I=1,C
43      SUM=SUM+HV(I)*SV(J,I)
        F(J,CT)=F(J,CT)+SD1*SUM
        IF(ITYPER(MM).EQ.0)GOTO 45
        DO 44 I=1,C
44      F(J,CT)=F(J,CT)+HL(I)*(RL(JM1)*SL(JM1,I)+SD0*SL(J,I))
        GOTO 265
45      SUM=0.
        SUM1=0.
        CALL ENL(TJ,HV)
        DO 46 I=1,C
        SUM=SUM+HV(I)*SL(J,I)
46      SUM1=SUM1+HL(I)*SL(JM1,I)
        F(J,CT)=F(J,CT)+SD0*SUM+RL(JM1)*SUM1
        GOTO 265
!-------------------------------------------------------------
! REFLUX RATIO SPECIFIED
50      DO 51 MM=1,NCOND
        IF(J.EQ.JCOND(MM))GOTO 52
51      CONTINUE
        TYPE 910,J
910     FORMAT(1X,'---ERROR---'/1X,'STAGE ',I2,' IS NOT A CONDENSER')
        STOP
52      F(J,CT)=BL(J)-BV(J)*SPCVAL(KK)
        GOTO 265
!-------------------------------------------------------------
! REBOILER RATIO (V/B) SPECIFIED
55      DO 56 MM=1,NREBL
        IF(J.EQ.JREBL(MM))GOTO 57
56      CONTINUE
        TYPE 911,J
911     FORMAT(1X,'---ERROR---'/1X,'STAGE ',I2,' IS NOT A REBOILER')
        STOP
57      F(J,CT)=BV(J)-BL(J)*SPCVAL(KK)
        GOTO 265
!-------------------------------------------------------------
! TEMPERATURE OF THE STAGE IS SPECIFIED
60      F(J,CT)=T(J)-SPCVAL(KK)
        GOTO 265
!-------------------------------------------------------------
! TOTAL VAPOR-FLOW V(J) LEAVING STAGE J IS SPECIFIED
! [ OR "D" IN CASE OF A CONDENSER ]
!-------------------------------------------------------------
65      F(J,CT)=BV(J)-SPCVAL(KK)
        GOTO 265
!-------------------------------------------------------------
! TOTAL LIQUID-FLOW L(J) LEAVING STAGE J IS SPECIFIED
! [ OR "B" IN CASE OF A REBOILER ]
!-------------------------------------------------------------
70      F(J,CT)=BL(J)-SPCVAL(KK)
        GOTO 265
!-------------------------------------------------------------
! COMPONENT-FLOW(VAPOR) V(j,i) OF COMPONENT i LEAVING STAGE j SPECIFIED
! [ OR D(i) IN CASE OF CONDENSER ]
75      II=ICOMP(KK)
        F(J,CT)=SV(J,II)-SPCVAL(KK)
        GOTO 265
!-------------------------------------------------------------
! COMPONENT-FLOW(LIQUID) L(j,i) OF COMPONENT i LEAVING STAGE j SPECIFIED
! [ OR B(i) IN CASE OF REBOILER ]
80      II=ICOMP(KK)
        F(J,CT)=SL(J,II)-SPCVAL(KK)
        GOTO 265
!-------------------------------------------------------------
! VAPOR-MOLE-FRACTION y(j,i) OF COMPONENT i LEAVING STAGE j SPECIFIED
! [ OR Xd(i) IN CASE OF CONDENSER ]
!-------------------------------------------------------------------
85      II=ICOMP(KK)
        F(J,CT)=SV(J,II)-SPCVAL(KK)*BV(J)
        GOTO 265
!-------------------------------------------------------------------
! LIQUID-MOLE-FRACTION X(J,1) OF COMPONENT i LEAVING STAGE j SPECIFIED
! [ OR Xb(i) IN CASE OF REBOILER ]
90      II=ICOMP(KK)
        F(J,CT)=SL(J,II)-SPCVAL(KK)*BL(J)
!--------------------------------------------------------------------------
265     KK=KK+1
        IF(KK.LE.NSPC)GOTO 270
        CONTINUE
!-----------------------------------------------------------------------
        DO 400 J=1,N
400     WRITE(51,*)J,(F(J,I),I=1,CT)
        WRITE(51,*)(QQ(J),J=1,N)
        WRITE(52,*)(RV(J),J=1,N)
        WRITE(52,*)(RL(J),J=1,N)
        WRITE(52,*)(DENOM(J),J=1,N)
!------------------------------------------------------------------------
        IZ1=47
        IZ2=57
        BL0=200.
        RL(1)=200./BL(20)
        RL(41)=1.-RL(1)
!------------------------------------------------------------------------
        J=1
        CALL TRIB(J)
        CALL TRIC(J)
        CALL OFFC(J,20,T(20))
        CALL INVPRT(B)
        CALL BCMUL(J)
        CALL BFMUL(J)
        CALL BAMUL(47,B,COFF)
!------------------------------------------------------------------------
        DO 501 JJ=2,9
        J=JJ
        JZ=IZ1+JJ-1
        CALL TRIA(J)
        CALL TRIB(J)
        CALL TRIC(J)
        CALL APVMUL(J-1)
        CALL AQMUL(J-1)
        DO 500 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 500 K=1,CP1
500     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        CALL INVPRT(B)
        CALL BCMUL(J)
        CALL BFMUL(J)
        CALL APLMUL(JZ-1)
        CALL MBAPL(JZ)
501     CONTINUE
!------------------------------------------------------------------------
        J=10
        JZ=56
        CALL ABSBOT
        CALL TRIA(J)
        CALL TRIB(J)
        CALL APVMUL(J-1)
        CALL AQMUL(J-1)
        DO 502 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 502 K=1,CP1
502     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        CALL BFMUL(J)
        CALL APLMUL(JZ-1)
        CALL MBAPL(JZ)
!----------------------------------------------------------------------
        J=11
        JZ=57
        CALL ABSTOP
        CALL TRIB(J)
        CALL TRIC(J)
        CALL INVPRT(B)
        CALL BCMUL(J)
        CALL BFMUL(J)
        CALL OFFA(J,47,T(47))
        CALL BAMUL(57,B,COFF)
!----------------------------------------------------------------------
        DO 504 JJ=12,19
        J=JJ
        JZ=JZ+1
        CALL TRIA(J)
        CALL TRIB(J)
        CALL TRIC(J)
        CALL APVMUL(J-1)
        CALL AQMUL(J-1)
        DO 506 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 506 K=1,CP1
506     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        CALL INVPRT(B)
        CALL BCMUL(J)
        CALL BFMUL(J)
        CALL APLMUL(JZ-1)
        CALL MBAPL(JZ)
504     CONTINUE
!----------------------------------------------------------------------
        J=20
        JZ=66
        CALL ABSBOT
        CALL TRIA(J)
        CALL TRIB(J)
        CALL APVMUL(J-1)
        CALL AQMUL(J-1)
        DO 507 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 507 K=1,CP1
507     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        DO 511 I=1,CT
        BETA1(I)=Q(10,I)
        DO 511 K=1,CP1
511     BETA2(I,K)=P(56,I,K)
        DO 508 KKK=9,1,-1
        K=KKK
        CALL PVPL(K)
        DO 509 I=1,CT
        DO 509 L=1,CP1
509     BETA2(I,L)=P(IZ1+K-1,I,L)-ALFA(I,L)
        CALL PVQMUL
        DO 510 I=1,CT
510     BETA1(I)=Q(K,I)-VEC1(I)
508     CONTINUE
        CALL OFFC(J,1,T(1))
        CALL CPLMUL(J,COFF,BETA2,ALFA)
        CALL CQMUL(J)
        DO 512 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 512 K=1,CP1
512     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        CALL INVPRT(B)
        CALL APLMUL(JZ-1)
        CALL BFMUL(J)
```

```
      CALL AQMUL(J-1)
      DO 542 I=1,CT
      F(J,I)=F(J,I)-VEC1(I)
      DO 542 K=1,CP1
542   B(I,K+C)=B(I,K+C)-ALFA(I,K)
      CALL INVPRT(B)
      CALL BFMUL(J)
      CALL APLMUL(JZ-1)
      CALL MBAPL(JZ)
!------------------------------------------------------------------
      J=36
      CALL CONBC(J,0)
      CALL INVPRT(B)
      CALL BCMUL(J)
      CALL BFMUL(J)
!------------------------------------------------------------------
      DO 543 JJ=37,39
      J=JJ
      CALL TRIA(J)
      CALL TRIB(J)
      CALL TRIC(J)
      CALL APVMUL(J-1)
      CALL AQMUL(J-1)
      DO 544 I=1,CT
      F(J,I)=F(J,I)-VEC1(I)
      DO 544 K=1,CP1
544   B(I,K+C)=B(I,K+C)-ALFA(I,K)
      CALL BCMUL(J)
      CALL BFMUL(J)
543   CONTINUE
!------------------------------------------------------------------
      J=40
      JZ=JZ+1
      CALL TRIA(J)
      CALL TRIB(J)
      CALL TRIC(J)
      CALL APVMUL(J-1)
      CALL AQMUL(J-1)
      DO 545 I=1,CT
      F(J,I)=F(J,I)-VEC1(I)
      DO 545 K=1,CP1
545   B(I,K+C)=B(I,K+C)-ALFA(I,K)
      CALL INVPRT(B)
      CALL BCMUL(J)
      DO 546 I=1,CT
546   BETA1(I)=Q(35,I)
      DO 547 KKK=34,21,-1
      K=KKK
      CALL PVQMUL
      DO 548 I=1,CT
548   BETA1(I)=Q(K,I)-VEC1(I)
547   CONTINUE
      DO 549 I=1,CT
      DO 549 K=1,CP1
549   BETA2(I,K)=P(79,I,K)
      DO 550 KKK=34,23,-1
      K=KKK
      KZ=79-K
      CALL PVPL(K)
      DO 551 I=1,CT
      DO 551 L=1,CP1
551   BETA2(I,L)=P(KZ,I,L)-ALFA(I,L)
550   CONTINUE
      CALL PVPV(21,22)
      CALL OFFC(J,21,T(21))
      CALL CPLMUL(J,COFF,BETA2,ALFA)
      CALL CQMUL(J)
      DO 552 I=1,CT
```

```
552     F(J,I)=F(J,I)-VEC1(I)
        CALL MBAPL(JZ)
        CALL BCMUL(J)
        CALL BFMUL(J)
!-----------------------------------------------------------------------
        J=41
        JZ=81
        CALL TRIA(J)
        CALL TRIB(J)
        CALL TRIC(J)
        CALL OFFA(J,20,T(20))
        CALL APVMUL(J-1)
        CALL AQMUL(J-1)
        DO 553 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 553 K=1,CP1
553     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        CALL INVPRT(B)
        CALL BCMUL(J)
        CALL AQMUL(J-1)
        DO 554 I=1,CT
554     F(J,I)=F(J,I)-VEC1(I)
        CALL BFMUL(J)
!       multiply CC=A.P(80)
!       multiply ALFA = AUFF.P(66)
        DO 555 I=1,CT
        DO 555 K=1,CP1
555     ALFA(I,K)=CC(I,K+C)+ALFA(I,K)
        CALL MBAPL(JZ)
!-----------------------------------------------------------------------
        JZ=81
        DO 556 JJ=42,45
        J=JJ
        JZ=JZ+1
        CALL TRIA(J)
        CALL TRIB(J)
        CALL TRIC(J)
        CALL APVMUL(J-1)
        CALL AQMUL(J-1)
        DO 557 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 557 K=1,CP1
557     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        CALL INVPRT(B)
        CALL BCMUL(J)
        CALL BFMUL(J)
        CALL APLMUL(JZ-1)
        CALL MBAPL(JZ)
556     CONTINUE
!-----------------------------------------------------------------------
        J=46
        JZ=85
        CALL TRIA(J)
        CALL TRIB(J)
        CALL TRIC(J)
        CALL APVMUL(J-1)
        CALL AQMUL(J-1)
        DO 558 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 558 K=1,CP1
558     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        CALL INVPRT(B)
        CALL BFMUL(J)
        CALL APLMUL(JZ-1)
!        subtraction CC=CC-ALFA
        CALL MTMUL2(B,CC)
!        multiply P(46)=B.CC
!-----------------------------------------------------------------------
```

```
        J=47
        CALL REBAB(J,0)
        CALL APVMUL(J-1)
        CALL AQMUL(J-1)
        DO 564 I=1,CT
        F(J,I)=F(J,I)-VEC1(I)
        DO 564 K=1,CP1
564     B(I,K+C)=B(I,K+C)-ALFA(I,K)
        CALL BFMUL(J)
!------------------------------------------------------------------
! BACK-SUBSTITUTION
!------------------------------------------------------------------
        J=46
        CALL QMPVX(J)
        JZ=85
        DO 570 JJ=45,40,-1
        J=JJ
        CALL QMPVX(J)
        CALL QMPLX(J,JZ,47)
        JZ=JZ-1
570     CONTINUE
        DO 571 JJ=39,36,-1
        J=JJ
        CALL QMPVX(J)
571     CONTINUE
        JZ=79
        CALL QMPLX(J,JZ,47)
        DO 572 JJ=34,23,-1
        J=JJ
        JZ=JZ-1
        CALL QMPVX(J)
        CALL QMPLX(J,JZ,47)
572     CONTINUE
        JZ=67
        DO 573 JJ=22,21,-1
        J=JJ
        CALL QMPVX(J)
573     CONTINUE
        JZ=66
        CALL QMPLX(J,JZ,47)
        DO 574 JJ=19,11,-1
        J=JJ
        JZ=JZ-1
        CALL QMPVX(J)
        CALL QMPLX(J,JZ,47)
574     CONTINUE
        IY1=20
        JZ=56
        CALL QMPLX(J,JZ,20)
        DO 575 JJ=9,1,-1
        J=JJ
        JZ=JZ-1
        CALL QMPVX(J)
        CALL QMPLX(J,JZ,20)
575     CONTINUE
!------------------------------------------------------------------
        STOP
        END
        SUBROUTINE ENV(TT,HV)
        DIMENSION HV(2)
        HV(1)=1.
        HV(2)=1.
        RETURN
        END
        SUBROUTINE ENL(TT,HL)
        DIMENSION HL(2)
        HL(1)=1.
        HL(2)=1.
```

```
      RETURN
      END
      SUBROUTINE FINDK(TT,AK)
      DIMENSION AK(2)
      AK(1)=1.
      AK(2)=1.
      RETURN
      END
!-----------------------------------------------------------------------
      SUBROUTINE CONBC(J,ITYPE)
!SUBROUTINE FOR COMPUTING THE ELEMENTS OF "B" & "C" SUBMATRICES FOR
!A PARTIAL OR TOTAL CONDENSER;ITYPE=0 FOR PARTIAL , ITYPE=1 FOR TOTAL
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XB/B(9,9)/XC/CC(9,9)
      COMMON /XVAR/SL(47,4),BL(47),RL(47),SV(47,4),BV(47),RV(47),T(47)
      COMMON /XCONST/ETA(47),SSL(47),SSV(47)
      DIMENSION XK(4,4),YK(4,4),AK(4)
      DO 5 I=1,CT
      DO 5 K=1,CT
5     CC(I,K)=0.
      GL=-(1.+SSL(J))
      GV=-(1.+SSV(J))
      EBYL=ETA(J)/BL(J)
      DO 10 I=1,C
      CC(I,I+C)=RV(J+1)
      B(I,CT)=0.
      B(I,I)=GL
10    B(I,I+C)=GV
      IF(ITYPE.EQ.1)GOTO 100
      CALL FINDK(T(J),AK)
      CALL DKBYDX(T(J),XK)
      CALL DKBYDY(T(J),YK)
      DO 20 I=1,C
      E1=-SL(J,I)*AK(I)/BL(J)
      E2=SV(J,I)/BV(J)
      DO 20 K=1,C
      B(I+C,K)=E1
      B(I+C,K+C)=E2
      IF(I.NE.K)GOTO 11
      B(I+C,K)=B(I+C,K)+AK(I)
      B(I+C,K+C)=B(I+C,K+C)-1.
11    SUMX=0.
      SUMY=0.
      DO 30 IP=1,C
      DELPK=0.
      IF(K.EQ.IP)DELPK=1.
      SUMX=SUMX+XK(I,IP)*(DELPK-SL(J,IP)/BL(J))
30    SUMY=SUMY+YK(I,IP)*(DELPK-SV(J,IP)/BV(J))
      B(I+C,K)=EBYL*(SL(J,I)*SUMX/BL(J)+B(I+C,K))
20    B(I+C,K+C)=(EBYL*SL(J,I)*SUMY+B(I+C,K+C))/BV(J)
      CALL DKBYDT(T(J),AK)
      DO 70 I=1,C
70    B(I+C,CT)=EBYL*SL(J,I)*AK(I)
      IF(ETA(J).EQ.1.)RETURN
      FAC1=(1.-ETA(J))/(BV(J+1)*BV(J+1))
      DO 80 I=1,C
      DO 80 K=1,C
      DELIK=0.
      IF(I.EQ.K)DELIK=1.
80    CC(I+C,K+C)=FAC1*(DELIK*BV(J+1)-SV(J+1,I))
      RETURN
!-----------------------------------------------------------------------
!FOR TOTAL CONDENSERS
100   SQV=1./(BV(J)*BV(J))
      SQL=1./(BL(J)*BL(J))
      DO 105 I=1,C
      E1=-SL(J,I)*SQL
```

```
          E2=SV(J,I)*SQV
          DO 105 K=1,C
          IF(I.EQ.K)GOTO 102
          B(I+C,K)=E1
          B(I+C,K+C)=E2
          GOTO 105
102       B(I+C,K)=(BL(J)-SL(J,I))*SQL
          B(I+C,K+C)=(SV(J,I)-BV(J))*SQV
105       CONTINUE
          RETURN
          END
!------------------------------------------------------------------------
          SUBROUTINE REBAB(J,ITYPE)
!SUBROUTINE FOR COMPUTING THE ELEMENTS OF "A" & "B" SUBMATRICES FOR
!A PARTIAL OR TOTAL REBOILER;ITYPE=0 FOR PARTIAL , ITYPE=1 FOR TOTAL
          INTEGER C,CP1,TWOC,CT
          COMMON N,C,CP1,TWOC,CT
          COMMON /XA/A(9,9)/XB/B(9,9)
          COMMON /XVAR/SL(47,4),BL(47),RL(47),SV(47,4),BV(47),RV(47),T(47)
          COMMON /XCONST/ETA(47),SSL(47),SSV(47)
          DIMENSION AK(4),XK(4,4),YK(4,4)
          GV=-(1.+SSV(J))
          GL=-(1.+SSL(J))
          DO 10 I=1,CT
          DO 10 K=1,CT
10        A(I,K)=0.
          DO 20 I=1,C
          A(I,I)=RL(J-1)
          B(I,I)=GL
          B(I,I+C)=GV
20        B(I,CT)=0.
          IF(ITYPE.EQ.1)GOTO 50
          CALL FINDK(T(J),AK)
          CALL DKBYDX(T(J),XK)
          CALL DKBYDY(T(J),YK)
          DO 30 I=1,C
          E1=-SL(J,I)*AK(I)/BL(J)
          E2=SV(J,I)/BV(J)
          DO 30 K=1,C
          B(I+C,K)=E1
          B(I+C,K+C)=E2
          IF(I.NE.K)GOTO 24
          B(I+C,K)=B(I+C,K)+AK(I)
          B(I+C,K+C)=B(I+C,K+C)-1.
24        SUMX=0.
          SUMY=0.
          DO 27 IP=1,C
          DELPK=0.
          IF(IP.EQ.K)DELPK=1.
          SUMX=SUMX+XK(I,IP)*(DELPK-SL(J,IP)/BL(J))
27        SUMY=SUMY+YK(I,IP)*(DELPK-SV(J,IP)/BV(J))
          B(I+C,K)=(SL(J,I)*SUMX/BL(J)+B(I+C,K))/BL(J)
          B(I+C,K+C)=(SL(J,I)*SUMY/BL(J)+B(I+C,K+C))/BV(J)
30        CONTINUE
          CALL DKBYDT(T(J),AK)
          DO 40 I=1,C
40        B(I+C,CT)=SL(J,I)*AK(I)/BL(J)
          RETURN
50        SQL=1./(BL(J)*BL(J))
          SQV=1./(BV(J)*BV(J))
          DO 60 I=1,C
          E1=-SL(J,I)*SQL
          E2=SV(J,I)*SQV
          DO 60 K=1,C
          IF(I.EQ.K)GOTO 51
          B(I+C,K)=E1
          B(I+C,K+C)=E2
          GOTO 60
```

```
51      B(I+C,K)=(BL(J)-SL(J,I))*SQL
        B(I+C,K+C)=(SV(J,I)-BV(J))*SQV
60      CONTINUE
        RETURN
        END
!----------------------------------------------------------------
        SUBROUTINE TRIA(J)
!SUBROUTINE TO COMPUTE THE ELEMENTS OF THE TRIDIAGONAL "A" MATRICES
        INTEGER C,CP1,TWOC,CT
        COMMON N,C,CP1,TWOC,CT
        COMMON /XA/A(9,9)/XCONST/ETA(47),SSL(47),SSV(47)
        COMMON /XVAR/SL(47,4),BL(47),RL(47),SV(47,4),BV(47),RV(47),T(47)
        DIMENSION HL(4),DHL(4)
        DO 10 I=1,CT
        DO 10 K=1,CT
10      A(I,K)=0.
        CALL ENL(T(J-1),HL)
        CALL ENV(T(J-1),DHL)
        SUM=0.
        DO 12 I=1,C
        A(I,I)=RL(J-1)
        A(CT,I)=RL(J-1)*HL(I)
12      SUM=SUM+SL(J-1,I)*DHL(I)
        A(CT,CT)=RL(J-1)*SUM
        RETURN
        END
!----------------------------------------------------------------
        SUBROUTINE TRIB(J)
!SUBROUTINE TO COMPUTE ELEMENTS OF THE TRIDIAGONAL "B" SUBMATRICES
        INTEGER C,CP1,TWOC,CT
        COMMON /XB/B(9,9)/XCONST/ETA(47),SSL(47),SSV(47)
        COMMON /XVAR/SL(47,4),BL(47),RL(47),SV(47,4),BV(47),RV(47),T(47)
        DIMENSION AK(4),XK(4,4),YK(4,4),HL(4),HV(4),DHL(4),DHV(4)
        S1=-(1.+SSL(J))
        S2=-(1.+SSV(J))
        DO 10 I=1,C
        B(I,I)=S1
        B(I,I+C)=S2
10      B(I,CT)=0.
        EBYL=ETA(J)/BL(J)
        COEFF=EBYL/BL(J)
        CALL FINDK(T(J),AK)
        CALL DKBYDX(T(J),XK)
        CALL DKBYDY(T(J),YK)
        DO 20 I=1,C
        E1=-AK(I)*SL(J,I)
        E2=SV(J,I)/BV(J)
        DO 20 K=1,C
        B(I+C,K)=E1
        B(I+C,K+C)=E2
        IF(I.NE.K)GOTO 21
        B(I+C,K)=B(I+C,K)+AK(I)*BL(J)
        B(I+C,K+C)=B(I+C,K+C)-1.
21      SUMX=0.
        SUMY=0.
!IF "K(I)" VALUES ARE COMPOSITION -INDEPENDENT THEN REMOVE THE "!"
!       GOTO 24
        DO 24 IP=1,C
        DELPK=0.
        IF(K.EQ.IP)DELPK=1.
        SUMX=SUMX+XK(I,IP)*(DELPK-SL(J,IP)*BL(J))
        SUMY=SUMY+YK(I,IP)*(DELPK-SV(J,IP)*BV(J))
24      CONTINUE
        B(I+C,K)=COEFF*(SL(J,I)*SUMX+B(I+C,K))
20      B(I+C,K+C)=(EBYL*SL(J,I)*SUMY+B(I+C,K+C))/BV(J)
        CALL DKBYDT(T(J),AK)
        DO 30 I=1,C
30      B(I+C,CT)=EBYL*SL(J,I)*AK(I)
```

```
        CALL ENL(T(J),HL)
        CALL ENV(T(J),HV)
        CALL DHLDT(T(J),DHL)
        CALL DHVDT(T(J),DHV)
        SUMX=0.
        SUMY=0.
        DO 33 I=1,C
        B(CT,I)=S1*HL(I)
        B(CT,I+C)=S2*HV(I)
        SUMX=SUMX+SL(J,I)*DHL(I)
33      SUMY=SUMY+SV(J,I)*DHV(I)
        B(CT,CT)=S1*SUMX+S2*SUMY
        RETURN
        END
!-----------------------------------------------------------------------
        SUBROUTINE TRIC(J)
!SUBROUTINE TTO COMPUTE THE ELEMENTS OF THE TRIDIAGONAL "C" MATRICES
        INTEGER C,CP1,TWOC,CT
        COMMON N,C,CP1,TWOC,CT
        COMMON /XC/CC(9,9)/XCONST/ETA(47),SSL(47),SSV(47)
        COMMON /XVAR/SL(47,4),BL(47),RL(47),SV(47,4),BV(47),RV(47),T(47)
        COMMON /XLINK/DENOM(47),NOFF,IX(6),IY(6),LORV(6),RATIO(6)
        DIMENSION VECTOR(4),HV(4),DHV(4)
        DO 10 I=1,CT
        DO 10 K=1,CT
10      CC(I,K)=0.
        CALL ENV(T(J+1),HV)
        CALL DHVDT(T(J+1),DHV)
        SUM=0.
        DO 11 I=1,C
        CC(I,I+C)=RV(J+1)
        CC(CT,I+C)=RV(J+1)*HV(I)
11      SUM=SUM+SV(J+1,I)*DHV(I)
        CC(CT,CT)=RV(J+1)*SUM
        IF(ABS(1.-ETA(J)).LE.1.E-6)RETURN
        LNK=0
        DO 12 I=1,C
12      VECTOR(I)=RV(J+1)*SV(J+1,I)
        DO 20 KK=1,NOFF
        IF((J.EQ.IX(KK)).AND.(LORV(KK).EQ.1))GOTO 15
        GOTO 20
15      LNK=1
        LL=IY(KK)
        DO 16 I=1,C
16      VECTOR(I)=VECTOR(I)+RATIO(LL)*SV(LL,I)
20      CONTINUE
        IF(LNK.EQ.1)GOTO 30
        COEFF=(1.-ETA(J))/(BV(J+1)*BV(J+1))
        DO 21 I=1,C
        E0=-COEFF*SV(J+1,I)
        E1=COEFF*(BV(J+1)-SV(J+1,I))
        DO 21 K=1,C
        IF(I.EQ.K)GOTO 22
        CC(I+C,K+C)=E0
        GOTO 21
22      CC(I+C,K+C)=E1
21      CONTINUE
        RETURN
30      COEFF=(1.-ETA(J))*RV(J+1)/(DENOM(J)*DENOM(J))
        DO 31 I=1,C
        E0=-COEFF*VECTOR(I)
        E1=COEFF*(DENOM(J)-VECTOR(I))
        DO 31 K=1,C
        IF(I.EQ.K)GOTO 32
        CC(I+C,K+C)=E0
        GOTO 31
32      CC(I+C,K+C)=E1
31      CONTINUE
```

```
        RETURN
        END
!-----------------------------------------------------------------------
        SUBROUTINE OFFA(J,JEXIT,TT)
!SUBROUTINE TO COMPUTE THE ELEMENTS OF THE OFF-DIAGONAL "A" MATRICES
!(HAVING STRUCTURE OF "A"-SUBMATRICES)
! *** Only for Liquid-Interlinks ***
! J     : stage no. into which the interlink-stream is entering
! JEXIT : stage no. from which the interlink-stream is leaving
! TT    : temperature of the interlink-stream entering stage-J
!         (TT may be equal to that of the leaving stage-JEXIT
!          or TT may be specified)
!
        INTEGER C,CP1,TWOC,CT
        COMMON N,C,CP1,TWOC,CT
        COMMON /XAOFF/AOFF(9,9)
        COMMON /XLINK/DENUM(47),NOFF,IX(6),IY(6),LORV(6),RATIO(6)
        COMMON /XVAR/SL(47,4),BL(47),RL(47),SV(47,4),BV(47),RV(47),T(47)
        DIMENSION HL(4),DHL(4)
        DO 10 I=1,CT
        DO 10 K=1,CT
10      AOFF(I,K)=0.
        DO 20 KK=1,NOFF
        IF((J.EQ.IX(KK)).AND.(JEXIT.EQ.IY(KK)))GOTO 30
20      CONTINUE
        TYPE 900,JEXIT,J
900     FORMAT(1X,'There is no stream from stage',I2,' to stage',I2)
        STOP
30      IF(LORV(KK).EQ.0)GOTO 40
        TYPE 901,J
901     FORMAT(1X,'subroutine OFFA is strictly for liquid interlinks
       1 and should not be called for vapor streams. Check stage',I2)
        STOP
40      CALL ENL(TT,HL)
        DO 50 I=1,C
        AOFF(I,I)=RATIO(KK)
50      AOFF(CT,I)=RATIO(KK)*HL(I)
        IF(ABS(TT-T(JEXIT)).GT.1.E-6)RETURN
        CALL DHLDT(TT,DHL)
        DO 70 I=1,C
70      AOFF(CT,CT)=AOFF(CT,CT)+SL(JEXIT,I)*DHL(I)
        AOFF(CT,CT)=AOFF(CT,CT)*RATIO(KK)
        RETURN
        END
!-----------------------------------------------------------------------
        SUBROUTINE OFFC(J,JEXIT,TT)
!SUBROUTINE TO COMPUTE THE ELEMENTS OF THE OFF-DIAGONAL "C" MATRICES
!(HAVING STRUCTURE OF "C"-SUBMATRICES)
! *** Only for Vapor-Interlinks ***
! J     : stage no. into which the interlink-stream is entering
! JEXIT : stage no. from which the interlink-stream is leaving
! TT    : temperature of the interlink-stream entering stage-J
!         (TT may be equal to that of the leaving stage-JEXIT
!          or TT may be specified)
!
        INTEGER C,CP1,TWOC,CT
        COMMON N,C,CP1,TWOC,CT
        COMMON /XCOFF/COFF(9,9)
        COMMON /XVAR/SL(47,4),BL(47),RL(47),SV(47,4),BV(47),RV(47),T(47)
        COMMON /XLINK/DENUM(47),NOFF,IX(6),IY(6),LORV(6),RATIO(6)
        DIMENSION HV(4),DHV(4),VECTOR(4)
        DO 10 I=1,CT
        DO 10 K=1,CT
10      COFF(I,K)=0.
        DO 20 KK=1,NOFF
        IF((J.EQ.IX(KK)).AND.(JEXIT.EQ.IY(KK)))GOTO 30
20      CONTINUE
        TYPE 900,JEXIT,J
900     FORMAT(1X,'There is no stream from stage',I2,' to stage ',I2)
        STOP
```

```
30      IF(LORV(KK).EQ.1)GOTO 40
        TYPE 901,J
901     FORMAT(1X,'Subroutine OFFC is strictly for vapor interlinks
     1  and should not be called for liquid streams. Check stage',I2)
        STOP
40      CALL ENV(TT,HV)
        DO 50 I=1,C
        COFF(I,I+C)=RATIO(KK)
50      COFF(CT,I+C)=RATIO(KK)*HV(I)
        IF(ABS(TT-T(JEXIT)).GT.1.E-6)GOTO 70
        CALL DHVDT(TT,DHV)
        DO 60 I=1,C
60      COFF(CT,CT)=COFF(CT,CT)+SV(JEXIT,I)*DHV(I)
        COFF(CT,CT)=RATIO(KK)*COFF(CT,CT)
70      IF(ABS(1.-ETA(J)).LE.1.E-6)RETURN
        DO 80 I=1,C
80      VECTOR(I)=RV(J+1)*SV(J+1,I)
        DO 90 LL=1,NOFF
        IF((J.EQ.IX(LL)).AND.(LORV(LL).EQ.1))GOTO 95
        GOTO 90
95      DO 99 I=1,C
99      VECTOR(I)=VECTOR(I)+RATIO(LL)*SV(IY(LL),I)
90      CONTINUE
        FAC1=(1.-ETA(J))*RATIO(KK)/DENOM(J)
        DO 100 I=1,C
        E0=-FAC1*VECTOR(I)/DENOM(J)
        E1=FAC1*(1.-VECTOR(I)/DENOM(J))
        DO 100 K=1,C
        IF(I.EQ.K)GOTO 110
        COFF(I+C,K+C)=E0
        GOTO 100
110     COFF(I+C,K+C)=E1
100     CONTINUE
        RETURN
        END
!-----------------------------------------------------------------------
        SUBROUTINE PROB(J,LABEL,ITYPE,VAL,ICOM)
!This subroutine computes the last row of the "A","B" & "C" submatrices,
!depending on the type of specification variable(LABEL) for any stage J
! J      : stage no.
! ITYPE  : if stage J is a condenser or a reboiler
!          then set ITYPE = 0 for a "partial" condenser or reboiler
!          else assign any integer to ITYPE
! VAL    : if no specification-value is required
!          then assign any real number to VAL
!          else assign given value of the specification variable to VAL
! ICOM   : component no.(if not required then assign any integer)
        INTEGER C,CP1,TWOC,CT
        COMMON N,C,CP1,TWOC,CT
        COMMON /XA/A(9,9)/XB/B(9,9)/XC/CC(9,9)
        COMMON /XVAR/SL(47,4),BL(47),RL(47),SV(47,4),BV(47),RV(47),T(47)
        COMMON /XCONST/ETA(47),SSL(47),SSV(47)
        DIMENSION HV(4),HVP1(4),HL(4),HLM1(4),DHV(4),DHL(4)
        GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130)LABEL
        TYPE 900,J,LABEL
900     FORMAT(1X,'error in specification,for stage',I3,'spec. type',I3)
        STOP
! NO CONDENSER SPECIFICATIONS
10      S1=-(1.+SSL(J))
        S2=-(1.+SSV(J))
        IF(ITYPE.EQ.1)GOTO 15
        CALL ENV(T(J),HV)
        CALL ENV(T(J+1),HVP1)
        CALL DHVDT(T(J),DHV)
        SUM=0.
        DO 12 I=1,C
        B(CT,I)=S1*HVP1(I)
        B(CT,I+C)=S2*HV(I)
```

```
12        SUM=SUM+SV(J,I)*DHV(I)
          B(CT,CT)=S2*SUM
          CALL DHVDT(T(J+1),DHV)
          CC(CT,CT)=0.
          DO 13 I=1,C
          CC(CT,I)=0.
          CC(CT,I+C)=RV(J+1)*HVP1(I)
13        CC(CT,CT)=CC(CT,CT)+(RV(J+1)*SV(J+1,I)+S1*SL(J,I))*DHV(I)
          RETURN
15        CALL ENL(T(J),HL)
          CALL DHLDT(T(J),DHL)
          B(CT,CT)=0.
          DO 16 I=1,C
          B(CT,I)=S1*HL(I)
          B(CT,I+C)=S2*HL(I)
          B(CT,CT)=B(CT,CT)+(RV(J+1)*SV(J+1,I)+S2*SV(J,I)+S1*SL(J,I))*
          1 DHL(I)
          CC(CT,I)=0.
16        CC(CT,I+C)=RV(J+1)*RV(J+1)*HL(I)
          CC(CT,CT)=0.
          RETURN
!NO REBOILER SPECIFICATIONS
20        S1=-(1.+SSL(J))
          S2=-(1.+SSV(J))
          IF(ITYPE.EQ.1)GOTO 25
          CALL ENL(T(J-1),HLM1)
          CALL ENL(T(J),HL)
          CALL DHLDT(T(J-1),DHL)
          A(CT,CT)=0.
          DO 22 I=1,C
          A(CT,I)=RL(J-1)*HLM1(I)
          A(CT,I+C)=0.
          A(CT,CT)=A(CT,CT)+(RL(J-1)*SL(J-1,I)+S2*SV(J,I))*DHL(I)
          B(CT,I)=S1*HL(I)
22        B(CT,I+C)=S2*HLM1(I)
          CALL DHLDT(T(J),DHL)
          SUM=0.
          DO 23 I=1,C
23        SUM=SUM+SL(J,I)*DHL(I)
          B(CT,CT)=S1*SUM
          RETURN
25        CALL ENL(T(J-1),HLM1)
          CALL DHLDT(T(J-1),DHL)
          A(CT,CT)=0.
          DO 26 I=1,C
          A(CT,I)=RL(J-1)*HLM1(I)
          A(CT,I)=0.
          A(CT,CT)=A(CT,CT)+(RL(J-1)*SL(J-1,I)+S1*SL(J,I)+S2*SV(J,I))*
          1 DHL(I)
          B(CT,I)=S1*HLM1(I)
26        B(CT,I+C)=S2*HLM1(I)
          B(CT,CT)=0.
          RETURN
! CONDENSER-HEAT-DUTY SPECIFIED
30        S1=-(1.+SSL(J))
          S2=-(1.+SSV(J))
          CALL ENV(T(J+1),HV)
          CALL DHVDT(T(J+1),DHV)
          SUM=0.
          DO 31 I=1,C
          CC(CT,I)=0.
          CC(CT,I+C)=RV(J+1)*HV(I)
31        SUM=SUM+SV(J+1,I)*DHV(I)
          CC(CT,CT)=RV(J+1)*SUM
          IF(ITYPE.EQ.1)GOTO 35
          CALL ENL(T(J),HL)
          CALL ENV(T(J),HV)
          CALL DHLDT(T(J),DHL)
```

```
       CALL DHVDT(T(J),DHV)
       E1=0.
       E2=0.
       DO 32 I=1,C
       B(CT,I)=S1*HL(I)
       B(CT,I+C)=S2*HV(I)
       E1=E1+SL(J,I)*DHL(I)
32     E2=E2+SV(J,I)*DHV(I)
       B(CT,CT)=S1*E1+S2*E2
       RETURN
35     CALL ENL(T(J),HL)
       CALL DHLDT(T(J),DHL)
       E1=0.
       E2=0.
       DO 36 I=1,C
       B(CT,I)=S1*HL(I)
       B(CT,I+C)=S2*HL(I)
       E1=E1+SL(J,I)*DHL(I)
36     E2=E2+SV(J,I)*DHL(I)
       B(CT,CT)=S1*E1+S2*E2
       RETURN
! REBOILER-HEAT-DUTY SPECIFIED
40     S1=-(1.+SSL(J))
       S2=-(1.+SSV(J))
       CALL ENL(T(J-1),HLM1)
       CALL DHLDT(T(J-1),DHL)
       SUM=0.
       DO 41 I=1,C
       A(CT,I)=RL(J-1)*HLM1(I)
       A(CT,I+C)=0.
41     SUM=SUM+SL(J-1,I)*DHL(I)
       A(CT,CT)=RL(J-1)*SUM
       IF(ITYPE.EQ.1)GOTO 45
       CALL ENL(T(J),HL)
       CALL ENV(T(J),HV)
       CALL DHLDT(T(J),DHL)
       CALL DHVDT(T(J),DHV)
       E1=0.
       E2=0.
       DO 42 I=1,C
       B(CT,I)=S1*HL(I)
       B(CT,I+C)=S2*HV(I)
       E1=E1+SL(J,I)*DHL(I)
42     E2=E2+SV(J,I)*DHV(I)
       B(CT,CT)=S1*E1+S2*E2
       RETURN
45     CALL ENV(T(J),HV)
       CALL DHVDT(T(J),DHV)
       SUM=0.
       DO 46 I=1,C
       B(CT,I)=S1*HLM1(I)
       B(CT,I+C)=S2*HV(I)
46     SUM=SUM+SV(J,I)*DHV(I)
       B(CT,CT)=S2*SUM
       RETURN
! REFLUX RATIO (L/D) SPECIFIED
50     DO 51 I=1,C
       B(CT,I)=1.
51     B(CT,I+C)=-VAL
       B(CT,CT)=0.
       DO 52 I=1,CT
52     CC(CT,I)=0.
       RETURN
! REBOILER RATIO SPECIFIED
60     DO 61 I=1,C
       B(CT,I)=-VAL
61     B(CT,I+C)=1.
       B(CT,CT)=0.
```

```
         DO 62 I=1,CT
62       A(CT,I)=0.
         RETURN
! T(j) SPECIFIED
70       DO 71 I=1,CT
         A(CT,I)=0.
         B(CT,I)=0.
71       CC(CT,I)=0.
         B(CT,CT)=1.
         RETURN
! V(j) SPECIFIED (or D specified for condensers)
80       DO 81 I=1,CT
         A(CT,I)=0.
81       CC(CT,I)=0.
         DO 82 I=1,C
         B(CT,I)=0.
82       B(CT,I+C)=1.
         B(CT,CT)=0.
         RETURN
! L(j) SPECIFIED (or B specified for reboilers)
90       DO 91 I=1,CT
         A(CT,I)=0.
91       CC(CT,I)=0.
         DO 92 I=1,C
         B(CT,I)=1.
92       B(CT,I+C)=0.
         B(CT,CT)=0.
         RETURN
! SV(j,i) SPECIFIED (or d(i) specified for condensers)
100      DO 101 I=1,CT
         A(CT,I)=0.
         B(CT,I)=0.
101      CC(CT,I)=0.
         B(CT,C+ICOM)=1.
         RETURN
! SL(j,i) SPECIFIED (or b(i) specified)
110      DO 111 I=1,CT
         A(CT,I)=0.
         B(CT,I)=0.
111      CC(CT,I)=0.
         B(CT,ICOM)=1.
         RETURN
! y(j,i) SPECIFIED (or xd(i) specified for condensers)
120      DO 121 I=1,CT
         A(CT,I)=0.
121      CC(CT,I)=0.
         DO 122 I=1,C
         B(CT,I)=0.
122      B(CT,I+C)=-VAL
         B(CT,C+ICOM)=B(CT,C+ICOM)+1.
         B(CT,CT)=0.
         RETURN
! x(j,i) SPECIFIED (or xb(i) specified for reboilers)
130      DO 131 I=1,CT
         A(CT,I)=0.
131      CC(CT,I)=0.
         DO 132 I=1,C
         B(CT,I)=-VAL
132      B(CT,I+C)=0.
         B(CT,ICOM)=B(CT,ICOM)+1.
         B(CT,CT)=0.
         RETURN
         END
!----------------------------------------------------------------------
         SUBROUTINE PVPL(JPV)
         COMMON N,C,CP1,TWUC,CT
         COMMON /XP/P(85,9,5)/XBETA2/BETA2(9,5)/XALFA/ALFA(9,5)
         DO 10 I=1,CT
```

```
      DO 10 K=1,CP1
      ALFA(I,K)=0.0
      DO 10 L=1,CP1
10    ALFA(I,K)=ALFA(I,K)+P(JPV,I,L)*BETA2(L+C,K)
      RETURN
      END
      SUBROUTINE PVPV(JPV1,JPV2)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XP/P(85,9,5)/XALFA/ALFA(9,5)
      DO 10 I=1,CT
      DO 10 K=1,CP1
      ALFA(I,K)=0.0
      DO 10 L=1,CP1
10    ALFA(I,K)=ALFA(I,K)+P(JPV1,I,L)*P(JPV2,L+C,K)
      RETURN
      END

      SUBROUTINE PLPV(U1,U2,U3)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      DIMENSION U1(9,5),U2(9,5),U3(9,5)
      DO 10 I=1,CT
      DO 10 K=1,CP1
      U3(I,J)=U1(I,CT)*U2(CT,K)
      DO 10 L=1,C
10    U3(I,K)=U3(I,K)+U1(I,L)*U2(L,K)
      RETURN
      END
      SUBROUTINE PLPL(U1,U2,U3)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      DIMENSION U1(9,5),U2(9,5),U3(9,5)
      DO 10 I=1,CT
      DO 10 K=1,CP1
      U3(I,J)=U1(I,CT)*U2(CT,K)
      DO 10 L=1,C
10    U3(I,K)=U3(I,K)+U1(I,L)*U2(L,K)
      RETURN
      END
      SUBROUTINE APVMUL(J)
                INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XA/A(9,9)/XP/P(85,9,5)/XALFA/ALFA(9,5)
      RRR=A(1,1)
      IF(ABS(RRR-1.0).LE.1.0E-06) GO TO 30
      DO 10 I=1,C
      DO 10 K=1,CP1
10    ALFA(I,K)=RRR*P(J,I,K)
      GO TO 40
30    DO 20 I=1,C
      DO 20 K=1,CP1
20    ALFA(I,K)=P(J,I,K)
40    DO 50 K=1,CP1
      ALFA(CT,K)=A(CT,CT)*P(J,CT,K)
      DO 50 L=1,C
50    ALFA(CT,K)=ALFA(CT,K)+A(CT,L)*P(J,L,K)
      RETURN
      END
      SUBROUTINE APLMUL(JZM1)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XA/A(9,9)/XP/P(85,9,5)/XALFA/ALFA(9,5)
      RRR=A(1,1)
      IF(ABS(1.0-RRR).LE.1.E-06) GO TO 20
      DO 10 I=1,C
      DO 10 K=1,CP1
```

```
   10     ALFA(I,K)=RRR*P(JZM1,I,K)
          GO TO 50

   20     DO 30 I=1,C
          DO 30 K=1,CP1
   30     ALFA(I,K)=P(JZM1,I,K)
   50     DO 60 K=1,CP1
          ALFA(CT,K)=A(CT,CT)*P(JZM1,CT,K)
          DO 60 L=1,C
   60     ALFA(CT,K)=ALFA(CT,K)+A(CT,L)*P(JZ,L,K)
          RETURN
          END
          SUBROUTINE CPVMUL(J,COFF,BETA2,ALFA)
          INTEGER C,CP1,TWOC,CT
          COMMON N,C,CP1,TWOC,CT
          DIMENSION COFF(9,9),BETA2(9,5),ALFA(9,5)
          RRR=COFF(1,CP1)
          IF(ABS(1.0-RRR).LE.1.0E-06) GO TO 30
          DO 10 I=1,C
          DO 10 K=1,CP1
   10     ALFA(I,K)=RRR*BETA2(I+C,K)
          GO TO 40

   30     DO 20 I=1,C
          DO 20 K=1,CP1
   20     ALFA(I,K)=BETA2(I+C,K)
   40     IF(ABS(1.0-ETA(J)).LE.1.0E-06) GO TO 70
          DO 50 I=CP1,TWOC
          DO 50 K=1,CP1
          ALFA(I,K)=0.0
          DO 50 L=CP1,TWOC
 50       ALFA(I,K)=ALFA(I,K)+COFF(I,L)*BETA2(L,K)
          GO TO 80
   70     DO 60 I=CP1,TWOC
          DO 60 K=1,CP1
   60     ALFA(I,K)=0.0
 80       DO 90 K=1,CP1
          ALFA(CT,K)=0.0
          DO 90 L=CP1,CT
   90     ALFA(CT,K)=ALFA(CT,K)+COFF(CT,L)*BETA2(L,K)
          RETURN
          END
          SUBROUTINE CPLMUL(J,COFF,BETA2,ALFA)
          INTEGER C,CP1,TWOC,CT
          COMMON N,C,CP1,TWOC,CT
          DIMENSION COFF(9,9),BETA2(9,5),ALFA(9,5)
          RRR=COFF(1,CP1)
          IF(ABS(1.0-RRR).LE.1.0E-06) GO TO 30
          DO 10 I=1,C
          DO 10 K=1,CP1
   10     ALFA(I,K)=RRR*BETA2(I+C,K)
          GO TO 40
   30     DO 20 I=1,C
          DO 20 K=1,CP1
   20     ALFA(I,K)=BETA2(I+C,K)
   40     IF(ABS(1.0-ETA(J)).LE.1.E-06) GO TO 70
          DO 50 I=CP1,TWOC
          DO 50 K=1,CP1
          ALFA(I,K)=0.0
          DO 50 L=CP1,TWOC
   50     ALFA(I,K)=ALFA(I,K)+COFF(I,L)*BETA2(L,K)
          GO TO 80
   70     DO 60 I=CP1,TWOC
          DO 60 K=1,CP1
   60     ALFA(I,K)=0.0
   80     DO 90 K=1,CP1
          ALFA(CT,K)=0.0
          DO 90 L=CP1,TWOC
 90      ALFA(CT,K)=ALFA(CT,K)+COFF(CT,L)*BETA2(L,K)
         RETURN
         END
         SUBROUTINE BCMUL(J)
         INTEGER C,CP1,TWOC,CT
         COMMON N,C,CP1,TWOC,CT
         COMMON /XB/B(9,9)/XCC/CC(9,9)/XP/P(85,9,5)
         RRR=CC(1,C+1)
         CTCT=CC(CT,CT)
         IF(ABS(1.0-ETA(J)).LE.1.E-06) GO TO 100
         IF(ABS(1.0-RRR).LE.1.E-06)GO TO 110
         DO 10 I=1,CT
         P(J,I,CP1)=B(I,CT)*CTCT
         DO 10 K=1,C
         P(J,I,K)=RRR*B(I,K)
         DO 10 L=CP1,CT
 10      P(J,I,K)=P(J,I,K)+B(I,L)*CC(L,K+C)
         RETURN
 110     DO 20 I=1,CT
         P(J,I,CP1)=B(I,CT)*CTCT
         DO 20 K=1,C
         P(J,I,K)=B(I,K)
         DO 20 L=CP1,CT
 20      P(J,I,K)=P(J,I,K)+B(I,L)*CC(L,K+C)
         RETURN
 100     IF(ABS(1.0-RRR).LE.1.E-06) GO TO 120
         DO 30 I=1,CT
         P(J,I,CP1)=B(I,CT)*CTCT
         DO 30 K=1,C
30       P(J,I,K)=RRR*B(I,K)
         RETURN
 120     DO 40 I=1,CT
         P(J,I,CT)=B(I,CT)*CTCT
         DO 40 K=1,C
 40      P(J,I,K)=B(I,K)
         RETURN
         END
         SUBROUTINE BAMUL(JZ,B,COFF)
         INTEGER C,CP1,TWOC,CT
         COMMON N,C,CP1,TWOC,CT
         COMMON /XP/P(85,9,5)

         DIMENSION COFF(9,9),B(9,9)
         RRR=COFF(1,1)
         CTCT=COFF(CT,CT)
         IF(ABS(1.0-RRR).LE.1.0E-06) GO TO 100
         DO 10 I=1,CT
         P(JZ,I,CP1)=B(I,CT)*CTCT
         DO 10 K=1,C
 10      P(JZ,I,K)=RRR*B(I,K)+B(I,CT)*COFF(CT,K)
         RETURN

 100     DO 20 I=1,CT
         P(JZ,I,CP1)=B(I,CT)*CTCT
         DO 20 K=1,C
 20      P(JZ,I,K)=B(I,K)+B(I,CT)*COFF(CT,K)
         RETURN
         END
         SUBROUTINE MBAPL(JZ)
         INTEGER C,CP1,TWOC,CT
         COMMON /XP/B(9,9)/XALFA/ALFA(9,5)/XP/P(85,9,5)
         DO 10 I=1,CT
         DO 10 K=1,CP1
         P(JZ,I,K)=-B(I,CT)*ALFA(CT,K)
         DO 10 L=1,C
 10      P(JZ,I,K)=P(JZ,I,K)-B(I,L)*ALFA(L,K)
         RETURN
         END
```

```
      SUBROUTINE MBAPV(JZ)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XB/B(9,9)/XALFA/ALFA(9,5)/XP/P(85,9,5)
      DO 10 I=1,CT
      DO 10 K=1,CP1
      P(JZ,I,K)=-B(I,CT)*ALFA(CT,K)
      DO 10 L=1,C
 10   P(JZ,I,K)=P(JZ,I,K)-B(I,L)*ALFA(L,K)
      RETURN
      END
      SUBROUTINE AQMUL(JM1)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XA/A(9,9)/XQ/Q(47,9)/XVEC1/VEC1(9)
      RRR=A(1,1)
      VEC1(CT)=A(CT,CT)*Q(JM1,CT)
      SUM=0.0
      IF(ABS(1.0-RRR).LE.1.E-06) GO TO 100
      DO 10 I=1,C
      VEC1(I)=RRR*Q(JM1,I)
      VEC1(I+C)=0.0
 10   SUM=SUM +A(CT,I)*Q(JM1,I)
      VEC1(CT)=VEC1(CT)+SUM
      RETURN
 100  DO 20 I=1,C
      VEC1(I)=Q(JM1,I)
      VEC1(I+C)=0.0
 20   SUM=SUM+A(CT,I)*Q(JM1,I)
      VEC1(CT)=VEC1(CT)+SUM
      RETURN
      END
      SUBROUTINE CQMUL(J)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XCOFF/COFF(9,9) /XBETA1/BETA1(9)/XVEC1/VEC1(9)
      RRR =COFF(1,CP1)
      VEC1(CT)=COFF(CT,CT)*BETA1(CT)
      IF(ABS(1.0-ETA(J)).LE.1.E-06) GO TO 100
      IF(ABS(1.0-RRR).LE.1.E-06) GO TO 50
      DO 10 I=1,C
      VEC1(I)=RRR*BETA1(I+C)
      VEC1(I+C)=0.0
      VEC1(CT)=VEC1(CT)+COFF(CT,I+C)*BETA1(I+C)
      DO 10 L=CP1,TWOC
 10   VEC1(I+C)=VEC1(I+C)+COFF(I+C,L)*BETA1(L)
      RETURN
 50   DO 20 I=1,C
      VEC1(I)=BETA1(I+C)
      VEC1(I+C)=0.0
      VEC1(CT)=VEC1(CT)+COFF(CT,I+C)*BETA1(I+C)
      DO 20 L=CP1,TWOC
 20   VEC1(I+C)=VEC1(I+C)+COFF(I+C,L)*BETA1(L)
      RETURN
 100  IF(ABS(1.0-RRR).LE.1.E-06) GO TO 150
      DO 30 I=1,C
      VEC1(I)=RRR*BETA1(I+C)
      VEC1(I+C)=0.0
 30   VEC1(CT)=VEC1(CT)+COFF(CT,I+C)*BETA1(I+C)
      RETURN
 150  DO 40 I=1,C
      VEC1(I)=BETA1(I+C)
      VEC1(I+C)=0.0
 40   VEC1(CT)=VEC1(CT)+COFF(CT,I+C)*BETA1(I+C)
      RETURN
      END
      SUBROUTINE BFMUL(J)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XB/B(9,9)/XF/F(85,9)/XQ/Q(85,9)
      DO 10 I=1,CT
      Q(J,I)=0.0
      DO 10 K=1,CT

 10   Q(J,I)=B(I,K)*F(J,K)
      RETURN
      END
      SUBROUTINE QMPVX(J)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XP/P(85,9,5)/XQ/Q(47,9)
      DO 10 I=1,CT
      DO 10 K=1,CP1

 10   Q(J,I)=Q(J,I)-P(J,I,K)*Q(J+1,K+C)
      RETURN
      END
      SUBROUTINE QMPLX(J,JP,JQ)
      INTEGER C,CP1,TWOC,CT
      COMMON N,C,CP1,TWOC,CT
      COMMON /XP/P(85,9,5)/XQ/Q(47,9)
      DO 10 I=1,CT
      Q(J,I)=Q(J,I)-P(JP,1,CP1)*Q(JQ,CT)
      DO 10 K=1,C
 10   Q(J,I)=Q(J,I)-P(JP,I,K)*Q(JQ,K)
      RETURN
      END
      SUBROUTINE INVPRT(B)
      INTEGER C,CT
      COMMON C,CT,N
      DIMENSION B(CT,CT),BS(4,4),BS1(4,4),PA(4),PB(4)
      DO 1 I=1,C
      PA(I)=1./B(I,I)
 1    CONTINUE
      DO 2 I=1,C
      DO 2 J=1,C
      BS(I,J)=PA(I)*B(I,J+C)
 2    CONTINUE
      DO 3 I=1,C
      DO 3 J=1,C
      DO 3 IJ=1,C
      BS1(I,J)=BS1(I,J)+B(I+C,IJ)*BS(IJ,J)
 3    CONTINUE
      DO 4 I=1,C
      DO 4 J=1,C
      BS1(I,J)=B(I+C,J+C)-BS1(I,J)
 4    CONTINUE
      CALL MATIN(BS1,C,XXX,0,DETERM)
      DO 5 I=1,C
      DO 5 J=1,C
      B(I+C,J+C)=BS1(I,J)
      BS1(I,J)=B(I+C;J)*PA(J)
      B(I,J+C)=0.
      B(I+C,J)=0.
 5    CONTINUE
      DO 6 I=1,C
      DO 6 J=1,C
      B(I,J)=0.
      DO 6 IJ=1,C
      B(I,J+C)=B(I,J+C)-BS(I,IJ)*B(IJ+C,J+C)
 6    CONTINUE
      DO 7 I=1,C
      DO 7 J=1,C
      BS(I,J)=0.
      DO 7 IJ=1,C
      B(I+C,J)=B(I+C,J)-B(I+C,C+IJ)*BS1(IJ,J)
      B(I,J)=B(I,J)-B(I,IJ+C)*BS1(IJ,J)
7     CONTINUE
      DO 9 I=1,C
      BS(I,I)=PA(I)
      PA(I)=0.
      DO 9 J=1,C
      B(I,J)=B(I,J)+BS(I,J)
9     CONTINUE
      DO 10 I=1,C
      DO 10 II=1,C
      PA(I)=PA(I)+B(I,II)*B(II,CT)+B(I,II+C)*B(II+C,CT)
      PB(I)=PB(I)+B(I+C,II)*B(II,CT)+B(I+C,II+C)*B(II+C,CT)
10    CONTINUE
      DO 11 I=1,C
       BB=BB+B(CT,I)*PA(I)+B(CT,I+C)*PB(I)
11    CONTINUE
      B(CT,CT)=1./(B(CT,CT)-BB)
      DO 12 I=1,C
      B(I,CT)=-PA(I)*B(CT,CT)
      B(I+C,CT)=-PB(I)*B(CT,CT)
      PA(I)=0.
      PB(I)=0.
12    CONTINUE
      DO 13 J=1,C
      DO 13 II=1,C
      PA(J)=PA(J)+B(CT,II)*B(II,J)+B(CT,II+C)*B(II+C,J)
      PB(J)=PB(J)+B(CT,II)*B(II,J+C)+B(CT,II+C)*B(II+C,J+C)
13    CONTINUE
      DO 14 I=1,C
      B(CT,I)=-B(CT,CT)*PA(I)
      B(CT,I+C)=-B(CT,CT)*PB(I)
14    CONTINUE
      DO 15 I=1,C
      DO 15 J=1,C
      B(I,J)=B(I,J)-B(I,CT)*PA(J)
      B(I,J+C)=B(I,J+C)-B(I,CT)*PB(J)
      B(I+C,J)=B(I+C,J)-B(I+C,CT)*PA(J)
      B(I+C,J+C)=B(I+C,J+C)-B(I+C,CT)*PB(J)
      BS1(I,J)=0.
15    CONTINUE
      BB=0.
      DO 22 I=1,C
      PB(I)=0.
22    CONTINUE
      RETURN
      END
      SUBROUTINE MATIN(A,N,B,M,DETERM)
C     A=CO-EFFICIENT OF ORDER N
C     B=VECTOR OF ORDER N
C     M=IF M IS SET TO ZERO,ONLY INVERSEIS COMPUTED
C     DETERM=VALUE OF DETERMENENT RETURNED
C     ------------------------------------------------
      DIMENSION A(N,N),B(N,1),IPIVOT(70),INDEX(70,2),DT(70)
      EQUIVALENCE (IROW,JROW),(ICOLUM,JCOLUM),(AMAX,T,SWAP)
C     INITIALIZATION
      DETERM=1.0
      DO 20 J=1,N
20    IPIVOT(J)=0
C     SEARCH FOR PIVOT ELEMENT
      DO 550 I=1,N
      AMAX=0.0
      DO 105 J=1,N
      IF(IPIVOT(J)-1)60,105,60
60    DO 100 K=1,N
      IF(IPIVOT(K)-1)80,100,740
80    IF(AMAX-ABS(A(J,K)))85,100,100
85    IROW=J
      ICOLUM=K
```

```
        AMAX=ABS(A(J,K))
 100    CONTINUE
 105    CONTINUE
        IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1
 C      INTRECHANGE ROWS TO PUT PIVOT VECTORS ONDIAGONAL
        IF(IROW-ICOLUM)140,260,140
 140    DETERM=-DETERM
        DO 200 L=1,N
        SWAP=A(IROW,L)
        A(IROW,L)=A(ICOLUM,L)
 200    A(ICOLUM,L)=SWAP
        IF(M)260,260,210
 210    DO  250 L=1,M
        SWAP=B(IROW,L)
        B(IROW,L)=B(ICOLUM,L)
 250    B(ICOLUM,L)=SWAP
 260    INDEX(I,1)=IROW
        INDEX(I,2)=ICOLUM
 C      DIVIDE PIVOT ROW BY PIVOR ELEMENT
        PIVOT=A(ICOLUM,ICOLUM)
        DT(I)=PIVOT
        A(ICOLUM,ICOLUM)=1.0
        DO 220 L=1,N
 220    A(ICOLUM,L)=A(ICOLUM,L)/PIVOT
 C      REDUCE NON PIVOT ROWS
        IF(M)380,380,360
 360    DO 370  L=1,M
 370    B(ICOLUM,L)=B(ICOLUM,L)/PIVOT
 380    DO 550 L1=1,N
        IF(L1-ICOLUM)400,550,400
 400    T=A(L1,ICOLUM)
        A(L1,ICOLUM)=0.0
        DO 450 L=1,N
 450    A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
        IF(M)550,550,460
 460    DO 500 L=1,M
 500    B(L1,L)=B(L1,L)-B(ICOLUM,L)*T
 550    CONTINUE
 C      INTERCHANGE THE COLUMNS
        DO 710 I=1,N
        L=N+1-I
        IF(INDEX(L,1)-INDEX(L,2))630,710,630
 630    JROW=INDEX(L,1)
        JCOLUM=INDEX(L,2)
        DO 705 K=1,N
        SWAP=A(K,JROW)
        A(K,JROW)=A(K,JCOLUM)
        A(K,JCOLUM)=SWAP
 705    CONTINUE
 710    CONTINUE
        DO 11 K=1,N
        IF(IPIVOT(K).NE.1)GO TO 12
 11     CONTINUE
        RETURN
 12     WRITE(22,991)
 991    FORMAT(/30X,"MATRIX IS SINGULAR"/)
 740    RETURN
        END
 C      ************************************************************************
```